Markt&Technik

Markus Breuer

AMIGA 500-BUCH

Kennenlernen und Anwenden der neuen Computer-Technologie:
★ Systemarchitektur ★ Workbench 1.2 ★ Intuition ★ CLI ★ Amiga-Grafik ★ Sound-Erzeugung ★ Amiga-BASIC ★ Schnittstellen

Amiga-500-Buch

Markus Breuer

AMIGA 500-Buch

Kennenlernen und Anwenden der neuen Computer-Technologie:
★ Systemarchitektur ★ Workbench 1.2
★ Intuition ★ CLI ★ Amiga-Grafik ★ Sound-Erzeugung
★ Amiga-BASIC ★ Schnittstellen

Markt&Technik Verlag AG

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Breuer, Markus
Amiga-500-Buch : Kennenlernen u. Anwenden d. neuen Computer-Technologie:
Systemarchitektur, Workbench 1.2, Intuition, CLI, Amiga-Grafik,
Sound-Erzeugung, Amiga-BASIC, Schnittstellen / Markus Breuer. –
Haar bei München : Markt-u.-Technik-Verl., 1987.
ISBN 3-89090-522-6

15 14 13 12 11 10 9 8 7 6 5 4 3 2
90 89 88 87

ISBN 3-89090-522-6

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis

Vorwort

Der Commodore Amiga 500 ist in mancher Hinsicht ein ganz besonderer Computer. Er beeindruckt vor allem durch zwei Dinge: eine qualitativ hochwertige, ungewöhnlich vielseitige und schnelle Farbgrafik und die Möglichkeit, mit mehreren Programmen gleichzeitig zu arbeiten (Multitasking). Beides erscheint auf den ersten Blick vielleicht als mehr oder weniger überflüssige Spielerei, wird aber zu einer nahezu unentbehrlichen und nur noch ungern wieder aufgegebenen Annehmlichkeit, wenn man erst einmal eine Weile damit gearbeitet hat. Lassen Sie sich nicht von anders lautenden Meinungen beirren! Grafik und Multitasking sind keine grundsätzlich notwendigen Bestandteile der Computer-Bedienung – schließlich sind die Computer Jahrzehnte ohne sie ausgekommen. Sie tragen beide aber ungemein zu einer einfach zu erlernenden, schnellen und angenehmen Bedienung eines Computers bei!

Im Amiga 500 werden auf allen Gebieten – sowohl bei der Hardware wie auch bei der Software – die allerneuesten Technologien eingesetzt. Für Sie als (potentiellen) Käufer dieses Computers bedeutet dies, daß Sie ungemein leistungsfähige Hardware zu einem vergleichsweise geringen Preis bekommen. (In der Computerbranche dürfte das allerdings nicht mehr allzu beeindruckend wirken – schon nächstes Jahr wird sicherlich ein noch leistungsfähigerer Computer für noch weniger Geld auf den Markt kommen.)

Die Hauptaufgabe dieses Buches ist es aber nicht, Sie mit den phantastischen Fähigkeiten des Amiga zu beeindrucken – selbst wenn sich das bisher vielleicht so angehört haben sollte. Vielmehr werden der Amiga und die Programme, die Sie brauchen, um ihn nutzbringend einzusetzen, exakt beschrieben. Dies soll Ihnen im täglichen Umgang mit dem Gerät und diesen Programmen helfen und Ihnen zugleich eine Entscheidungshilfe und Vergleichsmöglichkeiten mit anderen Computern geben, wenn Sie erst planen, sich (noch) einen anzuschaffen. (Direkte Vergleiche mit anderen Computern werden aber vermieden. Sie würden zu unfair ausfallen...)

Überblick über das Buch

Dieses Buch läßt sich grob in drei Teile gliedern, die auch im Inhalt immer wieder angesprochen werden. In jedem dieser Teile geht es ein wenig »tiefer« in die Materie hinein. Es ist deshalb nicht unbedingt sinnvoll oder gar nötig, sämtliche Kapitel in einem Zug durchzuarbeiten. Sie können auch immer erst dann mit einem neuen Kapitel beginnen, wenn Sie bei Ihrer Arbeit auf einen neuen Aspekt gestoßen sind, über den Sie mehr erfahren möchten. Das ist aber Geschmackssache! Wie Sie dieses Buch benutzen, bleibt letztlich Ihnen überlassen – hoffentlich nicht nur als Stütze für einen wackligen Tisch.

Im ersten Teil dieses Buches werden die wesentlichen Grundzüge der Bedienung des Amiga beschrieben. Sie sollten in diesem Teil mit der Philosophie des Amiga vertraut werden – seiner stark grafisch orientierten Bedienung, der Anwendung der Maus und aller wichtigen Handgriffe, die Sie später tagtäglich benötigen werden. Hierzu wird das Programm, das den Mittler zwischen Ihnen und der Hardware des Amiga spielt, die sogenannte *Workbench* (zu deutsch: *Werkbank*), ausführlich beschrieben. Am Ende dieses ersten Teils finden Sie dann auch Beschreibungen einiger wichtiger Anwendungsprogramme für den Amiga 500. Wenn Sie diese Programme noch nicht kennen, kann Ihnen dieses Kapitel die Kaufentscheidung hoffentlich leichter machen.

Der zweite Buch-Teil beschäftigt sich mit einer Art, den Amiga 500 zu bedienen, die nicht ganz so bildlich und intuitiv zu begreifen ist wie die Workbench: dem sogenannten *CLI*. Diese zweite Benutzerschnittstelle des Amiga hat trotz gewisser Nachteile und Schwierigkeiten, die sie dem Anfänger bereitet, auch einige Vorteile. Diese kommen vor allem dem intensiven Anwender zugute und denjenigen unter den Lesern, die mit dem kleinen Nagetier namens »Maus« auf Kriegsfuß stehen.

Der dritte Teil dieses Buches schließlich enthält vor allem Informationen für diejenigen Leser, die es immer ganz genau wissen wollen. Die Hardware des Amiga (und was man damit machen kann) wird beschrieben und ein erster Eindruck davon vermittelt, wie sich der Amiga dem Programmierer darbietet. Auch die Kontakte zur Außenwelt, Anschlüsse für Drucker und andere Zusatzgeräte, werden hier beschrieben. Die Programmierer unter den Lesern – und solche, die es werden wollen – lernen schließlich anhand von ein paar kleinen Beispielprogrammen in Amiga-BASIC, was man alles schon mit dieser einfachen und leicht erlernbaren Sprache mit dem Amiga 500 anstellen kann.

Was dieses Buch sein will – und was nicht

Ein einführendes Buch über einen Computer stellt zwangsläufig immer einen Kompromiß dar. Es muß ganz verschiedene Leserkreise, mit und ohne Computer-Vorwissen, mit und ohne Interesse für die Hardware und Programmierung, und mit und ohne den Computer, um den es geht, ansprechen. Ich habe versucht, für jeden dieser potentiellen Leserkreise etwas einzubringen, mußte dafür aber zwangsläufig in Kauf nehmen, diejenigen, die etwas mehr zu einem bestimmten Aspekt wissen wollen, zu enttäuschen. Leser, die bislang überhaupt nichts über

Computer wissen, werden deshalb in einigen Kapiteln im dritten Teil des Buches manchmal Schwierigkeiten haben, dem Text zu folgen.

Für diese beiden enttäuschten Gruppen ist Rettung aber nahe. Es gibt sowohl über die Computertechnik »an sich« einige sehr gute einführende Werke. Aber auch der an bestimmten Details interessierte Leser wird in anderen Büchern zu den Computern der Amiga-Familie, die speziell auf diese Details – wie Programmierung, Hardware und Grafik – eingehen, finden können, was er sucht. Diese Bücher sind bereits erhältlich und es erscheinen auch laufend neue. Der Markt&Technik-Verlag bietet zum Beispiel schon eine breite Palette von Büchern und Zeitschriften zum Amiga an.

In diesem Buch werden Sie aber zunächst einmal einen »Rundumschlag« finden, der alle Aspekte des Amiga 500 zunächst einmal streift. Ich hoffe, damit nicht nur dem frischgebackenen Amiga-500-Besitzer helfen zu können, sondern vielleicht auch dem einen oder anderen noch Unentschlossenen helfen zu können, eine Kaufentscheidung für oder gegen den Amiga 500 zu treffen.

Manche Kapitel beschäftigen sich aber trotzdem recht detailliert mit einigen besonders wichtigen Aspekten des Amiga und sind für den täglichen Einsatz bei der Arbeit am Gerät gedacht. So werden Sie in den folgenden Kapiteln vollständige Handbücher für den Umgang mit der Workbench und dem CLI finden, die kaum Fragen offenlassen dürften. Ich habe dabei versucht, vor allem die Themenkreise etwas mehr auszuarbeiten, die im Original-Handbuch, das zusammen mit dem Amiga 500 ausgeliefert wird, etwas stiefmütterlich behandelt werden.

Zur Vorgeschichte dieses Buches

Zum Abschluß dieses Vorwortes muß ich fairerweise aber auch noch erwähnen, daß das vorliegende Buch nicht hundertprozentig neu ist. Wer schon *Das Amiga-Handbuch* (M&T Nr. 90223) besitzt oder kennt, dem werden viele Passagen bekannt vorkommen. Die einzelnen Kapitel dieses Buches stellen fast alle Überarbeitungen der Kapitel von *Das Amiga-Handbuch* dar oder bauen auf diesem Buch auf. *Das Amiga-Handbuch* beschäftigt sich ausschließlich mit dem Amiga 1000 und erschien zu einem Zeitpunkt (im Frühjahr '86), als es den Amiga in Deutschland noch nicht einmal offiziell zu kaufen gab.

Seitdem hat sich einiges an der Amiga-Software geändert (unter anderem wurden die ersten Betriebssystemversionen 1.0 und 1.1 durch die weiterentwickelte Workbench 1.2 ersetzt). Diese Änderungen wurden natürlich berücksichtigt. Weiterhin ist der Amiga 500 natürlich auch in manch anderer Hinsicht ein anderer Computer als der Amiga 1000, wodurch sich weitere Änderungen ergaben. Und als ich gerade dabei war, die sich aus diesen beiden Gründen ergebenden Änderungen auszuführen, habe ich gleich einige Kapitel völlig neu geschrieben und an anderen Stellen die Reihenfolge der Darstellung geändert, wo ich mit meinem früheren »Schaffen« unzufrieden war.

Danksagungen

Welches Vorwort wäre komplett ohne Danksagungen? Dieses jedenfalls nicht. Ich möchte an dieser Stelle einer Reihe von Menschen danken, ohne die dieses Buch nicht zustande gekommen wäre – oder aber nicht in dieser Form.

Für einen nie versiegenden Strom an Informationen und vielfältige Unterstützung beim Kampf mit den Herstellern sowie die Versorgung mit Hard- und Software möchte ich Christine Baumann vom Markt&Technik-Buchverlag danken. Ihre regelmäßigen Telefonanrufe hielten mich trotz widriger Umstände bei der Stange. Ohne diesen ständigen Druck wäre dieses Buch mindestens einen Monat später fertig geworden. Bei Horst Brandl von der Happy-Redaktion möchte ich mich für seelischen Beistand und die Versorgung mit wichtiger Software bedanken, die mir vor allem bei der ersten Fassung dieses Buches (siehe oben) sehr geholfen hat. Weiterhin schulde ich auch noch Manfred Kohlen besonderen Dank. Einige Passagen im letzten Teil der ersten Fassung dieses Buches, sowie die ersten Skizzen für einige der darin enthaltenen Zeichnungen stammen von ihm. Inzwischen ist er ja auch selbst schon als Amiga-Buchautor zu einigem Ruhm gekommen.

Weiterhin möchte ich meinen Mitarbeitern von der *Gruppe Nimbus* danken, insbesondere Ursula Langenstroer, die die folgenden Seiten in ihre endgültige Form brachte, Gaby Kubitsch, die mit Adleraugen so manchen Tippfehler erspähte, und schließlich Jörg Mescher, der für die Erstellung der Fotos und die fachliche Überprüfung des Textes zuständig war.

Und last but not least auch ein Wort des Dankes an meine Freundin Maria. Sie hat sowohl die Arbeiten an der ersten Fassung dieses Buches (zum Amiga 1000) wie auch an dem hier vorliegenden Buch mit viel Geduld und Nachsicht begleitet und dürfte nun wissen, was das Wort »Computer-Witwe« bedeutet!

Es ist schwer, die Begeisterung in Worte zu fassen, die man erlebt, wenn man die ersten Stunden und Tage mit einem Amiga arbeitet. Ich hoffe, daß trotzdem ein wenig dieser Begeisterung in den folgenden Kapiteln wiederzufinden ist.

markus breuer

1 Vorhang auf: Der Amiga 500!

Der Commodore Amiga ist ein Vertreter einer neuen Computer-Generation. Diese neue Generation legt – im Vergleich zu ihren Vorgängern – wesentlich mehr Wert auf die einfache Bedienung des Computers als auf die »nackten Daten«. Man könnte einen Vergleich mit Autos ziehen und sagen, daß die Polsterung der Sitze, die Qualität der Stereo-Anlage und auch die Bremsen an erster Stelle kommen und irgendwann viel später erst die PS-(alias kW-)Leistung des Motors. Fairerweise muß man allerdings zugeben, daß eine solche Umorientierung bei den Computern erst in dem Moment möglich war, in dem nach Erledigung der notwendigen Rechen- und Steuerungsaufgaben noch genügend Computerleistung »übrigblieb«, die sich um solche Komfort-Aspekte kümmern kann.

Deshalb darf man auch die »nackten Daten« eines solchen Computers nicht vergessen, denn erst sie machen den Komfort möglich. In diesem Kapitel erhalten Sie deshalb zunächst einen kleinen Überblick über die Leistungsdaten des Amiga 500 und eher »äußerlichen Aspekte« seiner Erscheinung. Dies beginnt zunächst mit der Hardware und dann mit der Software, die Sie erhalten, wenn Sie einen Amiga 500 kaufen. Dieser Überblick führt Sie praktisch im Laufschritt durch die Welt des Amiga 500. Er wird Ihnen deshalb zwangsläufig etwas oberflächlich vorkommen. Aber keine Angst: Fast alle angeschnittenen Themen werden in den folgenden Kapiteln noch im Detail erläutert.

Das Wort »Amiga« ist spanisch und bedeutet in etwa »Freundin«. Genau das sollte dieser Computer im Idealfall auch für Sie werden (natürlich nur im platonischen Sinn). Wie auch im wirklichen Leben, kann Ihnen diese neue Freundin übrigens auch viel Ärger bereiten. Betrachten wir aber zunächst einmal ganz unvoreingenommen ihr äußeres Erscheinungsbild. Man soll zwar nicht zuviel auf Äußerlichkeiten geben; vielleicht ist es aber trotzdem ganz interessant für Sie.

Wenn Sie bereits längere Zeit einen Amiga besitzen, können Sie dabei wahrscheinlich wenig Neues lernen. Sie sollten die folgenden Abschnitte trotzdem überfliegen, weil einige Begriffe eingeführt werden, die auch im restlichen Buch immer wieder gebraucht werden.

1.1 Äußerlichkeiten

Äußerlich macht der Amiga 500 einen recht attraktiven Eindruck. Das »Basispaket«, das natürlich ausgebaut werden kann (siehe unten), besteht aus drei Komponenten:

- Einer Systemeinheit mit eingebautem Diskettenlaufwerk
- Einer Stromversorgung (Netzteil)
- Einer Maus

Optional kann (und sollte!) man auch noch ein zusätzliches Diskettenlaufwerk erwerben, das außen an die Haupteinheit angeschlossen wird. Unbedingt notwendig zur Arbeit mit dem Amiga 500 ist ferner ein Monitor oder Fernsehgerät. Das folgende Bild zeigt eine Arbeitskonfiguration aus Amiga 500, Maus und Monitor.

Bild 1.1: *Der Amiga 500*

Die einzelnen Komponenten dieses Computersystems sind über Kabel miteinander verbunden. Die Maus wird zum Beispiel über ein langes Kabel rechts an der Rückseite der Systemeinheit angeschlossen. Der Bildschirm steht normalerweise hinter der Haupteinheit (sie ist nicht groß genug, als daß er darauf Platz hätte) und ist über ein Kabel mit ihr verbunden. Besonders wichtig ist auch noch das Netzteil, das Sie am besten hinter dem Bildschirm oder auf dem Fußboden plazieren. Falls Sie sich für den Fußboden entscheiden – wo es am wenigsten stört – müssen Sie sich allerdings bei jedem Einschalten des Amiga bücken. Ein genialer Commodore-Ingenieur hat nämlich den Netzschalter in das Netzteil verlegt.

Die Länge der verschiedenen Kabel läßt eine fast beliebige Positionierung der Komponenten zu. Ob Sie die Maus zum Beispiel lieber rechts oder links neben der Tastatur liegen haben wollen, bleibt Ihnen überlassen. Dicht neben der Tastatur sollte sie aber auf alle Fälle liegen (mehr dazu weiter unten).

Die Tastatur des Amiga ist – anders als Sie es vielleicht schon bei anderen Computern gesehen haben – fest mit der Systemeinheit verbunden, aber trotzdem angenehm flach. Sie enthält neben dem normalen Satz Schreibmaschinentasten einen separaten Zehnerblock für die schnelle und komfortable Zahleneingabe, einen umgekehrt T-förmigen Block mit Tasten zur Steuerung einer Schreibmarke (Cursor) und eine Reihe von zehn Funktionstasten am oberen Rand. Mit diesen Funktionstasten können Sie innerhalb von Programmen oft bestimmte wichtige Funktionen mit nur einem Tastendruck auslösen.

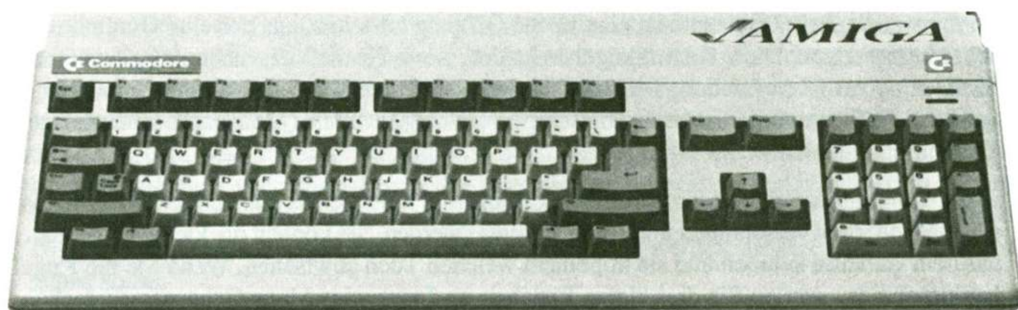


Foto Commodore

Bild 1.2: *Die Tastatur des Amiga*

Links und rechts neben der Leertaste liegen zwei besondere Tasten. Eine trägt das Commodore-Symbol ***C=***, die andere das Amiga-Markenzeichen ***A***. Es sind (in Kombination mit anderen Tasten) spezielle Funktionstasten, wie sie heute auch bei vielen anderen Rechnern zu finden sind (zum Beispiel Apple //, Apple Macintosh und Atari ST). Sie werden in den folgenden Kapiteln noch mehr über diese Tasten erfahren.

Direkt neben der Amiga- und Commodore-Taste liegt auf beiden Seiten noch eine sog. »Alt-Taste«. Hält man die Alt-Taste fest und drückt dann einen »normalen« Buchstaben, so erhält man besondere Zeichen, wie zum Beispiel á, œ, Σ und ¿. Viele Buchstabentasten sind mit einem solchen alternativen Symbol versehen, weshalb man den Amiga wohl zu Recht als einen

internationalen Computer bezeichnen kann. Fast jeder Buchstabe, den man sich vorstellen kann, steht mit der Alt-Taste zu Verfügung.

Der Anschlag der Tasten ist leicht und recht angenehm, sowohl für diejenigen, die nach dem Adler-System (Ziel suchen und dann hinabstoßen) die Buchstaben finden, als auch für die Zehn-Finger-Tipper. Der Tastenhub ist weder extrem lang noch besonders gering. Ein Buchstabe gilt allerdings schon als getippt, bevor man beim Anschlag ankommt. Beim Tippen jeder Taste hört man ein sanftes Klicken als Rückkopplung, daß der Buchstabe vom Rechner erkannt wurde. Dieses Klicken kommt nicht aus dem Lautsprecher, wie es bei einigen anderen Computern üblich ist, sondern ist mechanischen Ursprungs.

Die mechanische Maus des Amiga (es gibt auch optische Mäuse) besitzt eine elegante »Stromlinienform« und liegt gut in der Hand. An der Oberseite der Maus befinden sich zwei Knöpfe. Ein Drücken dieser Knöpfe wirkt in bestimmten Situationen als Befehl an den Computer. An ihrer Unterseite ist eine kleine Öffnung zu sehen, aus der eine Gummikugel etwas herausschaut. Diese Gummikugel dreht sich, wenn Sie die Maus über eine glatte Oberfläche schieben. Diese Drehung wird über das angeschlossene Kabel dem Computer mitgeteilt und kann für Steuerungszwecke am Bildschirm verwendet werden. (Mehr darüber im nächsten Kapitel.)

Die Kugel (und damit die Maus) rollt sehr flüssig, und falls Sie das einmal nicht mehr tut, ist sie mit Sicherheit verschmutzt und sollte gereinigt werden. Sie können die kleine Kugel dazu aus dem Gehäuse nehmen und sie mit einem weichen Tuch abwischen. Wenn Sie die Kugel herausnehmen, werden Sie drei kleine Rädchen im Innern des Maus-Gehäuses entdecken. Auch diese Rädchen müssen von Zeit zu Zeit gereinigt werden. Verwenden Sie dazu am besten ein Wattestäbchen, das Sie in etwas medizinischen Alkohol getaucht haben.

Das eingebaute Diskettenlaufwerk des Amiga 500 hat ein Fassungsvermögen (Kapazität) von 880 Kbyte, was ziemlich viel für ein Diskettenlaufwerk ist. 880 Kbyte sind mehr als 900 000 Buchstaben. Falls Sie nur Texte auf einer solchen Diskette speichern, so passen etwa 400 eineinhalbzeilig getippte Manuskriptseiten darauf. Die verwendeten Disketten mit 3,5 Zoll Durchmesser und Plastikschale sind handlich, robust und zuverlässig. Sie setzen sich jetzt allmählich als Standard bei neueren Mikrocomputern durch. Selbst IBM, der große »blaue Riese« des Computergeschäfts, setzt diese Disketten bei seiner neuesten Computerserie ein.

1.2 Innere Werte

Jetzt haben Sie schon die äußere Erscheinung des Amiga 500 etwas näher kennengelernt. Er besitzt aber auch innere Werte: Die Systemeinheit enthält unter anderem die folgenden Komponenten:

- Die CPU Motorola M68000
- 512 Kbyte RAM-Speicher für Programme und Daten
- 256 Kbyte ROM-Speicher für die wichtigsten Teile des Betriebssystems

- Ein 3,5-Zoll-Diskettenlaufwerk mit 880 Kbyte Kapazität
- Drei spezielle Chips für Sound und Grafik
- Diverse Zusatz-Chips für Hilfsaufgaben

Für Fachleute: Bei der CPU handelt es sich um den inzwischen recht weit verbreiteten Motorola-Prozessor 68000, der auch im Atari ST, dem Apple Macintosh und anderen leistungsfähigen PCs zu finden ist. Die CPU und der Speicher werden mit einer Frequenz von etwas unter 14,5 MHz getaktet. Die CPU selbst bekommt davon aber nur jeden zweiten Takt ab, so daß Sie effektiv nur mit der Hälfte dieser Taktfrequenz (ziemlich exakt 7,19 MHz) läuft. Die anderen Takte sind für die drei Spezial-Chips reserviert.

Mit den 256 Kbyte ROM-Speicher in der Haupteinheit des Amiga hat es eine besondere Bewandtnis. Wie die meisten 68000er-Computer wurde der Amiga von vornherein mit einem großen ROM entworfen. Dieses ROM sollte alle wesentlichen Teile des Betriebssystems, der Grafik- und Sound-Software sowie der Hilfsroutinen enthalten, die von den späteren Programmen genutzt werden können. Wie scheinbar allmählich üblich, gelang es den Software-Ingenieuren aber nicht, rechtzeitig zur Fertigstellung der Hardware auch die Software fertigzustellen. Man hat deshalb zunächst beim Amiga 1000 die ROMs aus dem Design herausgenommen und zusätzlichen RAM-Speicher an dessen Stelle eingebaut. Beim Einschalten des Amiga mußte dieser RAM-Speicher von der sogenannten »Kickstart-Diskette« mit der vorläufigen Version der Software gefüllt werden, die eigentlich in das ROM gehen sollte.

Bei den Nachfolgern des Amiga 1000, dem Amiga 2000 und dem Amiga 500, war die Software dann fertig und wurde wirklich in Form von ROM-Chips in die Computer eingebaut. Der Name »Kickstart« geistert auch heute noch durch die Amiga-Welt, und wenn Sie ihn einmal hören sollten, wissen Sie nun, wo er seinen Ursprung hat. (Das ROM des Amiga 500 wird in manchen Büchern und Zeitschriften zum Beispiel »Kickstart-ROM« genannt.)

1.3 Verbindungen zur Außenwelt

Der Amiga ist für die Kommunikation mit der Außenwelt – nicht nur mit dem Benutzer – gut gerüstet. An seiner Haupteinheit befinden sich reichlich Anschlüsse (sogenannte »Schnittstellen«), die fürs erste den meisten Anwendern völlig genügen dürften.

Hinten an der Haupteinheit befinden sich (von links nach rechts):

- Zwei Anschlüsse für je eine Maus oder einen Steuerknüppel (Joystick)
- Zwei Anschlüsse für Stereo-Tonausgabe
- Ein zusätzlicher Diskettenanschluß
- Eine serielle Schnittstelle (nach RS232-Standard)
- Eine parallele Schnittstelle (nach Centronics-Standard)

• Der Stromanschluß (Netzteil)

• Ein Anschluß für einen RGB-Monitor

• Ein Anschluß für Video-Monitore oder Fernsehbildschirme

Die wichtigsten dieser Anschlüsse dürften für Sie wahrscheinlich die parallele Schnittstelle sein, an die Sie eine Vielzahl handelsüblicher Drucker anschließen können, sowie die serielle Schnittstelle, an die Sie ebenfalls Drucker oder zum Beispiel ein Modem (für die Kommunikation mit anderen Computern über Telefonleitungen) anschließen können. Einer der Monitor-Ausgänge, der erste Mausanschluß und vielleicht auch der zusätzliche Diskettenlaufwerksanschluß, ist ja bei Ihnen wahrscheinlich schon belegt.

An der linken Seite befindet sich, versteckt hinter einer herausnehmbaren Plastikleiste, die »Erweiterungsschnittstelle«. Hier werden praktisch alle Signale, die intern im Amiga 500 erzeugt werden (für Fachleute: der komplette »Systembus«), nach außen gelegt. Hersteller von Zusatzgeräten, aber auch Commodore selbst, bieten bereits eine Reihe von Erweiterungen an, die Sie an dieser Stelle anschließen können, oder planen solche Erweiterungen.

An der Unterseite des Amiga 500 schließlich befindet sich, ebenfalls unter einer Plastikabdeckung verborgen, ein leeres Fach und eine Steckleiste, die speziell für Speichererweiterungen gedacht ist. Mehr dazu aber im folgenden Abschnitt.

1.4 Zubehör

Ähnlich wie bei einigen Automobilmarken der Luxusklasse, ist es meist mit dem Kauf des »Grundmodells« – und damit im übertragenen Sinn des Amiga 500 – allein nicht getan. Damit sind Sie zwar »fahrtüchtig«; viel Freude werden Sie an dieser Art zu fahren aber nicht haben. Deshalb ist es ganz sinnvoll, möglichst früh einen Blick in die Zubehörliste zu werfen, damit Sie wissen, was in dieser Hinsicht auf Sie zukommt. (Sie selbst sollten vielleicht auch einen Blick in die Zubehör-**Preisliste** werfen, damit Sie auch wissen, was auf Ihr Portemonnaie noch alles zukommt.)

1.4.1 Zusätzliche Diskettenlaufwerke

Der Amiga 500 enthält ein Diskettenlaufwerk. Dieses ist mit einer Kapazität von 880 Kbyte für den Anfang mehr als aureichend. Ein einziges Diskettenlaufwerk kann bei intensivem Gebrauch des Amiga 500 andererseits aber auch rasch »eng« werden. Insbesondere die Anfertigung von Diskettenkopien ist mit nur einem Laufwerk eine recht lästige Angelegenheit. Deshalb bietet der Amiga 500 die Möglichkeit, an dem Anschluß, der mit »Disk Port« beschriftet ist, ein zusätzliches Diskettenlaufwerk anzuschließen. Commodore bietet ein solches zusätzliches Diskettenlaufwerk für 3,5-Zoll-Disketten unter der Bezeichnung Amiga 1010 an. Dieses Laufwerk wird in einem separaten Gehäuse ausgeliefert, das vom Design und Farbton her zum Amiga 500 paßt. Es besitzt seinerseits auch wieder einen Anschluß für ein weiteres Diskettenlaufwerk. Auf diese Weise kann man mehrere zusätzliche

Diskettenlaufwerke an den Amiga 500 anschließen, obwohl nur ein Anschluß für diesen Zweck vorhanden ist.

Neben einem (oder mehreren) zusätzlichen 3,5-Zoll-Laufwerk(en) können aber auch Laufwerke für 5,25-Zoll-Disketten angeschlossen werden, die bis heute von vielen anderen Computerherstellern verwendet werden. Dieses Laufwerk wird von Commodore unter der Bezeichnung Amiga 1020 verkauft. Sie können dann auch auf den preiswerteren 5,25-Zoll-Disketten Daten und Programme speichern, müssen dafür aber auch mit einer geringeren Kapazität vorliebnehmen. (Auf eine 5,25-Zoll-Diskette passen nur etwas weniger als halb so viele Daten wie auf eine 3,5-Zoll-Diskette.) Ein 5,25-Zoll-Laufwerk versetzt den Amiga 500 aber auch – mit der entsprechenden Software – in die Lage, Daten von Disketten zu lesen, welche von anderen Computern darauf geschrieben wurden, die 5,25-Zoll-Laufwerke verwenden (zum Beispiel ein IBM PC).

1.4.2 Zusätzlicher RAM-Speicher

In einem vorangehenden Abschnitt wurde ja bereits erwähnt, daß an der Unterseite des Amiga 500 schon Platz für eine Speichererweiterung vorgesehen ist. Commodore bietet unter der Bezeichnung Amiga 501 eine solche Speichererweiterung an, die den RAM-Speicher des Amiga 500 von 512 Kbyte auf 1024 Kbyte (1 Mbyte) aufstockt. Sie werden dadurch in die Lage versetzt, größere und leistungsfähigere Programme zu benutzen, größere Datenmengen zu bearbeiten oder auch mit mehreren Programmen gleichzeitig zu arbeiten.

Mit der Speichererweiterung Amiga 501 bekommen Sie zugleich auch eine Uhr mit Batterieversorgung spendiert. Der Amiga 500 ist dann (nachdem Sie die Uhr einmal richtig gestellt haben) immer über die aktuelle Zeit im Bilde. Sie können ihn dann auch als Uhr, Wecker und Terminkalender »mißbrauchen«. Zugleich sorgt diese Uhr auch dafür, daß bei jeder Änderung, die Sie an einer Datei vornehmen, das Datum der Änderung mit abgespeichert wird. Dies ist eine unschätzbare Hilfe, wie Sie bei der täglichen Arbeit mit dem Amiga 500 bald merken werden.

Beides – Speichererweiterung und Uhr – sind außerordentlich sinnvolle Ergänzungen des Amiga 500. Man sollte die Speichererweiterung Amiga 501 deshalb am besten gleich beim Kauf des Grundgeräts miterwerben (vielleicht macht Ihnen Ihr Händler dann ja auch ein günstiges Sonderangebot).

1.4.3 Bildschirme

Ein wirklich unverzichtbares Zubehör des Amiga 500 ist ein Bildschirm. Hier stehen Ihnen sehr viele Möglichkeiten zur Wahl. Welche davon für Sie die beste ist, hängt von den Programmen, die Sie benutzen, Ihren Ansprüchen hinsichtlich der Bildqualität, und nicht zuletzt von Ihrem Geldbeutel ab.

Einen fast idealen Bildschirm für Ihren Amiga 500 stellt zweifellos der speziell für die verschiedenen Amiga-Modelle entwickelte RGB-Monitor dar, den die Firma Commodore zum Beispiel unter der Bezeichnung Amiga 1081 anbietet, und der von Philips produziert wird.

Ein »RGB-Monitor« – oder genauer Rot-Grün-Blau-Monitor – erhält die Signale für die drei Grundfarben, aus denen sich die Farbe eines Bildschirmpunktes zusammensetzt, über drei getrennte Kabel. Die Bildqualität, besonders die Farbtrennung und Schärfe, ist deshalb typischerweise wesentlich besser als bei einem sogenannten Composite-Monitor oder gar einem einfachen Farbfernseher. Der Amiga 1081 ist ein solcher (analoger) RGB-Monitor, der die volle Palette von 4096 Farben, zu der der Amiga fähig ist, darstellen kann. Neben seinen grafischen Fähigkeiten kann er aber auch 80spaltigen Text in akzeptabler Qualität zeigen. Das Bild ist stabil, flimmerfrei und die Farbtrennung ist bis in die Ecken hinein hervorragend (keine Regenbogen).

Die billigste – und leider auch von der Bildqualität her minderwertigste – Lösung des Bildschirmproblems ist es, einen Fernseher oder Video-Monitor zu verwenden. Hierzu benötigen Sie allerdings einen Fernseher mit Video-Eingang. (Der Amiga 500 besitzt keinen HF-Ausgang, den Sie mit dem Antenneneingang eines Fernsehers verbinden können.) Mit einem Fernseher erhalten Sie aber nur Schwarzweiß-Bilder (selbst wenn es ein Farbfernseher ist). Commodore hat bei früheren Amiga-Modellen eine Farbausgabe auf Farbfernsehern versucht, war aber mit der Qualität so unzufrieden, daß diese Möglichkeit im Amiga 500 nicht mehr vorgesehen wurde.

Wer einen guten Video-Monitor besitzt oder aber unbedingt Wert darauf legt, sich die Augen zu verderben, kann einen zusätzlichen Adapter oder »Modulator« namens Amiga 520 erwerben, der auf den RGB-Ausgang gesteckt wird. Dieser Modulator ermöglicht den Anschluß eines Farbfernsehers an den Amiga 500, der dann auch ein farbiges Bild liefert und sich nicht auf Schwarz und Weiß beschränkt. Auch für den Anschluß des Amiga 500 an einen Videorecorder – eine sehr interessante Verbindung übrigens – ist der Amiga-520-Adapter notwendig. Damit ist allerdings nicht ganz die Bildqualität des Amiga 1081 zu erreichen, wenngleich schon eine bessere Qualität als bei der Verwendung eines Fernsehers. Diese Alternative dürfte vor allem dann ideal sein, wenn man bereits einen solchen Monitor besitzt – zum Beispiel als Bestandteil einer Heim-Videoanlage. Preiswerter als der Amiga 1081 ist ein guter Video-Monitor nämlich nicht.

Wer bereits einen anderen IBM-kompatiblen Computer mit RGB-Monitor besitzt, von denen es eine große Auswahl gibt, kann auch diesen Monitor am Amiga 500 verwenden. Dann muß er sich allerdings meist mit 16 statt mit 4096 Farben begnügen (mehr schafft die herkömmliche Standard-IBM-Grafik nämlich nicht). Sehr gute Bildschärfe und alle 4096 Farben, zu denen der Amiga 500 fähig ist, kann man hingegen auch mit den sogenannten MultiSync- oder MultiScan-Monitoren erzielen, die normalerweise ebenfalls für IBM-kompatible Computer verkauft werden. (Sie werden meist im Zusammenhang mit den sogenannten »EGA-Karten« verwendet.) Solche Monitore sind aber meist deutlich teurer als der Amiga 1081 (sogar teurer als der Amiga 500 selbst). Ihre Anschaffung lohnt deshalb nur im Zusammenhang mit einem weiteren Computer oder bei besonders hohen Ansprüchen.

1.4.4 Drucker

Wer die Werke, die er mit seinem Computer zustande gebracht hat, nicht nur auf dem Bildschirm betrachten will, sondern sie auch schwarz (oder bunt) auf weiß in der Hand halten

möchte, wird nicht um die Anschaffung eines Druckers herumkommen. Hier ist die Auswahl ähnlich groß wie bei den Monitoren, das Problem andererseits aber nicht ganz so drängend, da man meist auch sehr gut – zumindest eine Zeitlang – ohne Drucker auskommen kann.

Commodore selbst bietet, speziell für die verschiedenen Amiga-Modelle, die beiden Drucker MPS 2000 und MPS 2000C an, die bei der Firma NEC gebaut werden. Bei beiden Geräten handelt es sich um grafikfähige Nadeldrucker der mittleren Preisklasse. Wesentlicher Unterschied zwischen den beiden Geräten ist nur, daß der MPS 2000C ein Mehrfarbdrucker ist, während der MPS 2000 immer nur in einer Farbe drucken kann (das »C« in der Bezeichnung des MPS 2000C steht für »Color«).

Daneben können Sie am Amiga 500 aber auch nahezu alle Computerdrucker verwenden, die im Handel angeboten werden. Die Software des Amiga unterstützt eine ungewöhnlich breite Palette von Druckern. Zwei wichtige Gesichtspunkte sind bei der Auswahl des Druckers aber zu beachten. Der erste davon ist die Grafikfähigkeit. Die Software des Amiga 500 ist stark grafikorientiert. Auf dem Bildschirm erscheinen nicht nur Zahlen und Buchstaben, sondern oft auch Symbole, Piktogramme, Skizzen und Bilder. Wenn Sie diese auch auf Papier bekommen wollen, brauchen Sie einen grafikfähigen Drucker. Fast alle heutzutage angebotenen Nadel- oder Matrixdrucker sind grafikfähig. Typenraddrucker, Kugelkopfdrucker oder zu Druckern umfunktionierte Schreibmaschinen sind jedoch nicht grafikfähig. Wenn Sie nicht absolut sicher sind, daß Sie immer nur Texte ausdrucken wollen, sollten Sie keinen solchen nichtgrafikfähigen Drucker kaufen.

Der zweite wichtige Gesichtspunkt ist die Farbfähigkeit des Druckers. Der Amiga 500 kann auf einem guten Monitor bis zu 4096 verschiedene Farben darstellen, während die meisten Drucker nur in einer Farbe drucken können. Wer die Bildschirmfarben auch zu Papier bringen will, muß also einen Mehrfarbdrucker kaufen. Diese Drucker haben jedoch zwei entscheidende Nachteile: sie sind entweder recht teuer oder bringen nur eine dürftige Qualität zu Papier und sind zudem noch recht langsam. Auch die laufenden Kosten – also die Kosten pro ausgedruckte Seite – sind bei Mehrfarbdruckern unverhältnismäßig viel höher als bei einfachen Matrixdruckern.

Falls Sie sich aber auch von diesen Tatsachen nicht abschrecken lassen, berücksichtigen Sie bitte auch noch, daß kein heute zu einem Preis unter 40 000 DM erhältlicher Drucker das zu Papier bringen kann, was Sie auf dem Bildschirm sehen. Die meisten Mehrfarbdrucker verfügen nur über etwa ein Dutzend unterschiedlicher Farben; ein Bruchteil dessen, was der Bildschirm zuwege bringt. Und den gedruckten Farben mangelt es meist auch an Leuchtkraft und Brillanz. Für den Privatbedarf kann ein Mehrfarbdrucker schon akzeptable Ergebnisse liefern. Für den betrieblichen Einsatz sind seine Erzeugnisse meist zu dürftig und entstammen gar zu deutlich dem Computer.

Ganz anders sieht die Situation bei der teuersten Druckerkategorie aus. Die besten Druckergebnisse kann man nämlich zweifellos mit einem Laserdrucker erzielen. Die Ausdrucke dieser etwa 5 000 DM bis 10 000 DM teuren Geräte stehen der Qualität einer Tageszeitung oder Illustrierten kaum nach. Sie sind zudem auch voll grafikfähig – bieten aber nur eine

Druckfarbe. Die Anschaffung eines Druckers, der fünfmal soviel kostet wie der Computer selbst, dürfte jedoch nur in den wenigsten Anwendungsfällen gerechtfertigt sein.

1.4.5 Sonstiges Zubehör

Neben diesem »üblichen« Zubehör, wie es Drucker und Bildschirme darstellen, gibt es für ganz spezielle Anwendungen natürlich auch noch ganz spezielles Zubehör. So gibt es zum Beispiel Zusatzprodukte, mit denen man Bilder, die man mit einer Videokamera aufgenommen hat, »in den Amiga 500 hineinbekommen kann« und sie dann verändern und wieder anzeigen und ausdrucken kann. Ähnliches ist mit Tönen möglich.

Wem 1 Mbyte RAM-Speicher nicht genügt, der kann seinem Amiga 500 auch noch mehr Speicher spendieren. Oder man kann noch höhere Kapazitäten und höhere Geschwindigkeiten als mit Disketten erreichen, indem man eine »Festplatte« an den Amiga 500 anschließt. Die Erweiterungsmöglichkeiten sind fast unbegrenzt. Dies wird vor allem durch die Erweiterungsschnittstelle ermöglicht, die sich hinter einer Plastikabdeckung links am Gehäuse des Amiga 500 befindet. Die Eigenschaften dieser Erweiterungsschnittstelle sind beim Amiga 500 übrigens nahezu dieselben wie beim Amiga 1000, bei dem sie allerdings auf der rechten Gehäuseseite lag. Einige Zusatzgeräte für den Amiga 1000 können deshalb auch am Amiga 500 verwendet werden. Die, bei denen das nicht geht, können von den Herstellern mit wenig Aufwand umkonstruiert werden, weshalb schon in Kürze mit einer wahren Flut von solchen Zusätzen und Erweiterungen zu rechnen ist.

1.5 Was Sie außer der Hardware noch bekommen

Damit soll aber die Betrachtung der Amiga-Hardware erst einmal abgeschlossen werden. In den folgenden Abschnitten wird die »weiche Ware« (Software) des Amiga 500 etwas näher vorgestellt. Neben der reinen Hardware bekommt man schon beim Kauf des Basispakets auch etwas Software dazu. Diese ist auf drei Disketten enthalten:

- Der Workbench-Diskette
- Der Extras-Diskette
- Der Erste-Schritte-Diskette

Die Workbench-Diskette ist die »Boot-Diskette«, mit der der Amiga 500 nach dem Start »hochgefahren« oder »gebootet« wird, wie die Computerleute sagen. Sie sorgt für das Erscheinen der grafischen Benutzerschnittstelle des Amiga 500, der »Workbench« oder zu deutsch Werkbank, mit der der Benutzer wohl die meiste Zeit zu tun haben wird. Sie enthält ansonsten noch einige kleine Hilfs- und Demonstrations-Programme, mit denen man die Möglichkeiten des Amiga 500 etwas austesten kann.

Die Extras-Diskette oder ExtrasD-Diskette enthält vor allem Amiga-BASIC. Amiga-BASIC ist eine sehr komfortable Variante des bekannten und eigentlich auf fast allen Mikrocomputern zu findenden Microsoft-BASIC, der am weitesten verbreiteten Computer-Software überhaupt.

Mehr dazu aber in einem späteren Kapitel. Zusätzlich zu der Programmiersprache selbst befinden sich auch noch einige Beispielprogramme samt den dazugehörigen Hilfsdateien und einige weitere kleine Programme auf der Extras-Diskette.

Die Erste-Schritte-Diskette schließlich enthält das Tutorial »Erste-Schritte«. Ein solches »Tutorial«, manchmal sagt man auch »Guided Tour« dazu, ist ein Programm, das Sie Schritt für Schritt durch eine Folge von Lernschritten führt. In diesem Fall erlernen Sie dabei die Grundzüge der Bedienung des Amiga. Sie brauchen diese Diskette nur in das Diskettenlaufwerk Ihres Amiga zu schieben, ihn einzuschalten und schon geht es los. Folgen Sie dann einfach den Anweisungen, die am Bildschirm erscheinen!

Neben diesen beiden Disketten erhalten die Commodore-Händler noch ein oder zwei Disketten mit zusätzlicher Demonstrations-Software. Die meisten Händler werden wohl auch nichts dagegen haben, ihren Kunden diese zu kopieren. Diese Demos zeigen vor allem, »was im Amiga 500 steckt«. Es sind viele phantastische Bilder, schnelle bewegte Grafiken, ein einfaches elektronisches Klavier und ähnliches. Zu den Demos gehört natürlich auch der schon legendäre »bouncing ball«, ein rotierender, auf und ab hüpfender, rot-weiß-karierter Globus, und der »Juggler«, ein Roboter, der drei silberne Bälle jongliert, in denen er und seine Umgebung sich spiegeln.

Mit dieser Grundausstattung werden Sie sich als frischgebackener Amiga-Besitzer aber wohl höchstens ein oder zwei Tage beschäftigen können, falls Sie nicht Amiga-BASIC benutzen wollen, um eigene Programme zu entwickeln. Für den Amiga-500-Anwender (Anwender hier im Gegensatz zum Programmierer, der sich seine eigenen Programme schreibt) gibt es bereits sehr viel zusätzliche Software – schließlich kann der Amiga 500 nahezu jedes Programm verarbeiten, das auf dem Amiga 1000 läuft, der schon seit fast zwei Jahren auf dem Markt ist. Von Commodore selbst können Sie zum Beispiel die Textverarbeitung TextCraft und das Malprogramm GrafiCraft erhalten. Viele andere Softwarehäuser bieten inzwischen auch Programme für fast jeden Anwendungszweck an. Die Vielfalt und Leistungsfähigkeit der Programme für den Amiga steht dem Softwareangebot für andere Computer kaum nach.

Außer den beiden eben genannten Disketten erhalten Sie beim Kauf natürlich auch noch Handbücher für die Bedienung Ihres neuen Computers, und zwar drei Stück. Eines behandelt den Amiga 500 (das Gerät) selbst und die Workbench, über die Sie weiter unten noch mehr erfahren werden. Dieses Handbuch ist recht kurz und knapp gefaßt und erklärt nur die wichtigsten Grundlagen der Bedienung. Es enthält aber auch wichtige Informationen über die Hardware des Amiga 500 und die Anschlüsse für Zusatzgeräte. Das zweite Handbuch beschäftigt sich mit dem CLI, über das Sie im zweiten Teil dieses Buches noch mehr erfahren werden, und das dritte schließlich mit Amiga-BASIC und den wichtigsten Grundzügen der Programmierung des Amiga 500. Neben einigen meist kleinen Beispielen enthält es vor allem ein sehr vollständiges Nachschlagewerk zu Amiga-BASIC.

1.6 Die Fähigkeiten des Amiga 500

Nachdem Sie nun – zumindest ungefähr – wissen, wie der Amiga 500 aussieht und was Sie alles erhalten, wenn Sie einen kaufen, sollen Sie nun aber auch einen kleinen Eindruck davon erhalten, wozu er fähig ist, wenn er die richtige Software »zu fressen bekommt«. Diese Leistungsdaten sind sehr beeindruckend – besonders dann, wenn Sie bereits andere Computer kennen, die meist nicht so leistungsfähig sind. Sie sollten sich davon aber nicht blenden lassen. Das wichtigste bei jedem Computer ist, daß er Ihnen Arbeit abnimmt. Dem Amiga 500 mit seinem schnellen 16-Bit-Hauptprozessor fällt das natürlich nicht schwer. Mit seinen phantastischen Grafik- und Sound-Fähigkeiten hat er aber dazu noch das Potential, Ihnen diese Arbeitserleichterung so angenehm und unterhaltsam wie möglich zu gestalten.

1.6.1 Grafik-Fähigkeiten

Der eine Aspekt, der am Amiga von allen Fachleuten nahezu uneingeschränkt gelobt wird, sind seine Grafikfähigkeiten. Sie sind den Möglichkeiten vergleichbarer Computer in vieler Hinsicht weit voraus. Drei speziell für diesen Zweck entworfene und von Commodore gebaute Chips (sogenannte »Custom-VLSI-Chips«) sorgen für eine Vielfalt an Möglichkeiten und vor allem für eine Geschwindigkeit beim Aufbau der Grafiken, wie man sie bisher – zumindest bei Computern in dieser Preislage – für unmöglich hielt.

Die Grafik des Amiga 500 ist bitmapped. Dies bedeutet, daß die Farbe jedes Punktes auf dem Bildschirm durch ein oder mehrere Bits im Speicher bestimmt wird. Im simpelsten Fall gibt es eine 1:1-Zuordnung, bei der jeder Punkt auf dem Bildschirm genau einem Bit im Speicher entspricht. Da der Amiga 500 mehrere Farben gleichzeitig in einer Grafik verwenden kann, liegt der Fall natürlich etwas komplizierter – aber im wesentlichen stimmt die 1:1-Zuordnung, von der auch der Begriff bitmapped abgeleitet ist.

Andere Computer haben manchmal noch einen zusätzlichen Betriebsmodus, in dem sie nur Text (Buchstaben, Zahlen und Sonderzeichen) auf dem Bildschirm darstellen können. Beim Amiga 500 gibt es einen solchen Modus nicht. Auch Text muß, wie ein Kreis oder ein Rechteck, als ein Punktmuster auf den Bildschirm gemalt werden. Dies ist natürlich schwieriger, aber dafür auch flexibler. Durch die Geschwindigkeit, die die speziellen Grafik-Chips dem Amiga 500 verschaffen, spielt der zusätzliche Aufwand zudem kaum eine Rolle.

Die Amiga-Grafik kennt allerdings auch mehrere Modi. Die horizontale Auflösung (Anzahl der Punkte in einer Bildzeile) kann 320 oder 640 Punkte betragen. Und die vertikale Auflösung (Anzahl der Bildzeilen beziehungsweise Anzahl der Punkte in einer Bildspalte) kann entweder 256 oder 512 Punkte betragen. Die vertikale 400-Punkte-Auflösung wird allerdings auf eine Art und Weise erreicht, die zu erhöhtem Bildschirmflimmern führt. Diese Methode nennt man »Interlaced Video«, sie wird auch bei Fernsehern angewandt, weshalb man nicht allzu lange nahe vor einem Fernseher sitzen kann, ohne Kopfschmerzen zu bekommen.

Die bitmapped Grafik des Amiga 500 ist mehrfarbig. Je nachdem, wieviel Speicher man opfern will, kann man zwischen 2, 4, 8, 16 und 32 möglichen Farben wählen, die gleichzeitig in einer Grafik erscheinen können. Bei mehr als 2 Farben wird auch mehr als 1 Bit pro Punkt benötigt.

Deshalb auch der erhöhte Gesamtspeicherbedarf für ein Bild mit mehreren Farben. Diese 32 Farben sind aber nicht fest »eingebaut«, sondern können aus einer Palette von 4096 Farben ausgewählt werden. Man sucht sich also »normalerweise« zunächst aus einer Palette von 4096 Farben 32 aus und kann mit diesen dann sein Bild malen.

Das Wort »normalerweise« im letzten Satz wurde mit Bedacht in Anführungszeichen gesetzt, weil es einen Betriebsmodus gibt, in dem – mit gewissen Einschränkungen allerdings – alle 4096 möglichen Farben gleichzeitig am Bildschirm erscheinen können. Dies ist der sogenannte *hold and modify*-Modus oder kurz »HAM-Modus«, über den Sie in einem folgenden Kapitel noch mehr erfahren werden. Zusätzlich ist es möglich, den Bildschirm in horizontale Streifen zu unterteilen, in denen verschiedene Modi gelten. Zum Beispiel kann die obere Hälfte oder das obere Drittel des Bildschirms 32 andere Farben verwenden als der untere Teil. Wie Sie sehen, sind die Grafikmöglichkeiten des Amiga 500 sehr vielseitig.

Bisher haben wir uns nur mit den statischen Aspekten der Amiga-Grafik beschäftigt, also ruhenden, unveränderlichen Bildern. Die Grafik-Chips des Amiga 500 bieten aber auch die Möglichkeit, 8 kleine Bilder (die sogenannten »Sprites«) mit hoher Geschwindigkeit über den Bildschirm zu bewegen. Dies ist nicht nur für Videospiele wichtig, wie Sie vielleicht zunächst meinen werden. Sprites können auch für durchaus sinnvolle Aufgaben verwendet werden.

Die Software des Amiga 500 bietet darüber hinaus aber in Zusammenarbeit mit anderen Teilen der Grafik-Chips auch noch weitergehende Möglichkeiten für solche bewegte Grafiken (der Fachmann spricht von »animierten« Grafiken oder von »Animation«), die dem Programmierer sehr viel Arbeit abnehmen. Hierzu gehören die BOBs (Blitter Objects) und AnimObjects (animierte Objekte).

Ein AnimObject kann zum Beispiel Bilder der verschiedenen Bewegungsphasen einer gehenden Person enthalten. Je mehr Phasen, das heißt, je mehr Speicher man verwendet, desto flüssiger wirkt diese Bewegung. Gibt man dem AnimObject dann eine Geschwindigkeit, dann könnte diese Person zum Beispiel langsam oder schnell über den Bildschirm schreiten. All dies geschieht – nach dem einmaligen »Einschalten« des Vorgangs – ohne weiteres Zutun des eigentlichen Programms. Dieses kann sich währenddessen um andere Dinge kümmern. Das bekannte Robocity-Demo, das sich ebenfalls auf den Demo-Disketten befindet, die jeder Amiga-Händler bekommt, nutzt exakt diese Möglichkeit. Zwei Roboter und ein Hund bewegen sich darin recht flüssig vor einem festen Hintergrund.

1.6.2 Sound-Fähigkeiten

Nicht ganz so große Publizität wie seine Grafikfähigkeiten haben die Möglichkeiten des Amiga 500 zur Klangerzeugung gefunden – unverdienterweise. Sie sind mindestens ebenso beeindruckend. Der Amiga 500 besitzt insgesamt 4 voneinander unabhängige Tonkanäle, von denen jeweils 2 zu einem Stereo-Kanal zusammengefaßt werden können. Jeder dieser 4 Kanäle kann eine völlig beliebige Wellenform mit beliebiger »Hüllkurve« ausgeben. Auf dem einen Kanal könnte zum Beispiel ein Klavier spielen, während auf dem anderen eine Trompete erklingt. Dazu wird nur in sehr geringem Maße der Prozessor beziehungsweise ein Eingreifen des Programms benötigt, das den Ton irgendwann einmal »angestoßen« hat.

Man darf von der Sound-Hardware des Amiga 500 natürlich nicht zuviel verlangen. Es handelt sich nicht um einen professionellen Synthesizer, und die Klangqualität kann Hi-Fi-Tests gewiß nicht standhalten. Für die Darstellung eines Klangs werden im Amiga 500 zum Beispiel 8 Bit breite Werte verwendet. Hi-Fi-Geräte in Digital-Technik verwenden üblicherweise 16 Bit, was zu einer rund 250mal besseren Klangqualität führt – von anderen Gesichtspunkten ganz abgesehen. Trotzdem können die vom Amiga 500 erzeugten Töne oft recht natürlich klingen.

Die vom Amiga 500 erzeugten Töne müssen aber nicht »natürlich« sein und klingen. Sie können auch vollständig »synthetisch« sein, also mehr oder weniger auf dem Papier beziehungsweise am Computer entworfen worden sein. Oder man kann auch einen natürlichen Klang nehmen, seine Charakteristika analysieren (ihn »digitalisieren«), abspeichern und dann von der Amiga-Sound-Hardware wieder ausgeben lassen. Dabei kann man den Klang in sehr weiten Grenzen ändern und »mißbrauchen«. Es wäre zum Beispiel durchaus möglich, Beethovens 5. Symphonie mit digitalisierten Kochtopfdeckeln zu spielen.

Das berühmte *bouncing-ball-Demo* nutzt zum Beispiel die Möglichkeit, einen Klang aufzunehmen und dann verändert wiederzugeben, in recht eindrucksvoller Weise. Der Ton, den man hört, wenn der Ball an eine Wand oder auf den Boden schlägt, ist in Wirklichkeit das Geräusch eines Mikrophons, das vom ehemaligen Amiga-Software-Chef Bob Pariseau vor ein Garagentor geschlagen wird.

Wie es inzwischen scheinbar schon guter Ton bei neuen Computern zu werden scheint, ist der Amiga 500 natürlich auch für die Kommunikation mit MIDI-Geräten (MIDI = Musical Instruments Digitital Interface; der wohl wichtigste Standard für die Verbindung digitaler Musikinstrumente, wie Synthesizer und Keyboards) vorbereitet. Hierzu muß ein entsprechendes Zusatzgerät an die serielle Schnittstelle des Amiga angeschlossen werden. Wem also die Klangqualität des Amiga auch dann noch nicht ausreicht, wenn er ihn an seine Stereoanlage angeschlossen hat, der sollte sich die Anschaffung des MIDI-Interfaces und eines entsprechenden MIDI-fähigen Synthesizers überlegen.

1.6.3 Sprach-Fähigkeiten

Eine Sound-Fähigkeit des Amiga 500 darf man auf keinen Fall vergessen: die Sprachausgabe. Der Amiga 1000 war seinerzeit der erste Computer, bei dem Software für Sprachsynthese sofort mitgeliefert wurde. Daran hat sich natürlich auch beim Amiga 500 nichts geändert. Sprachsynthese, also künstliche Sprache, kann in jedem Programm eingesetzt werden und ist völlig problemlos zu programmieren. In der simpelsten Form übergibt man einfach einen Text an eine bestimmte Routine, die ihn daraufhin in akustischer Form ausgibt. Die Aussprache kann man zusätzlich in verschiedenster Form beeinflussen. Die Stimmlage kann geändert werden; der Sprecher kann entweder im tiefen Baß oder als helle Frauenstimme sprechen. Die Geschwindigkeit der Sprache kann innerhalb weiter Grenzen geändert werden (in weiteren Grenzen als sinnvoll ist). Und schließlich kann die Stimme entweder sehr ausdrucksvoll, mit betonten Worten und Satzzeichen, oder monoton, computerhaft, sprechen.

Wir können also schon bald Programme erwarten, die uns unsere Fehler nicht nur am Bildschirm melden oder durch einen nervösen Summton mitteilen, sondern auch rufen: »Heh

du Dummkopf, was hast du da schon wieder angestellt?« Die Entscheidung, ob das einen echten Fortschritt darstellt, überlasse ich Ihnen.

1.6.4 Sound- und Grafik-Hardware

Nachdem Sie nun einen ersten Eindruck von den Grafik- und Sound-Fähigkeiten des Amiga 500 bekommen haben, möchte ich auch noch ganz kurz darauf eingehen, wie sie zustandekommen – die Details können Sie im letzten Teil dieses Buches nachlesen. Eine allzu genaue Kenntnis der Hardware-Zusammenhänge ist sowieso beim Amiga 500 nicht nötig (nicht einmal für die Programmierung), da einem die Amiga-Systemsoftware das meiste von der »Bit-Fummelei« abnimmt. Falls Sie allerdings ein Videospiel schreiben wollen, wie es die Welt noch nicht gesehen hat, werden Sie kaum um eine direkte Manipulation der Grafik-Chips herumkommen. Dann sind allerdings wirklich Sachen möglich, die »die Welt noch nicht gesehen hat«.

Die Qualität und Geschwindigkeit der Amiga-Grafik hat viele Gründe. Sie liegen aber im wesentlichen in den drei Custom-VLSI-Chips begründet, die wir schon mehrfach erwähnten. Zum Beispiel hat jeder der drei Zusatzchips im Amiga 500 ungefähr die Komplexität der CPU M68000, des Hauptprozessors im Amiga 500, der einer der modernsten und leistungsfähigsten Chips ist, die man heute in PCs einsetzt. Der Amiga 500 hat also – jedenfalls für bestimmte Aufgaben – ungefähr drei- bis viermal soviel Rechenleistung als ein Computer, der »nur« einen 68000er-Prozessor enthält. Die drei Chips tragen sinnigerweise die Spitznamen »Agnus«, »Denise« und »Paula« (obwohl gelegentlich auch andere Namen in der Computer-Presse zu lesen waren).

In diesen Chips befinden sich Zusatz-Prozessoren oder sogenannte »Koprozessoren«, die dem eigentlichen Mikroprozessor, dem M68000, viel Arbeit abnehmen. Sie arbeiten gleichzeitig »neben« diesem her, während er mit anderen Dingen beschäftigt ist. Einer dieser Koprozessoren ist zum Beispiel der sogenannte Blitter, der unter anderem sehr schnell Linien zeichnen, Flächen mit einer bestimmten Farbe füllen und beliebige Teilbilder von einem Ort am Bildschirm zu einer anderen Stelle bewegen oder auch entfernen kann.

Auch die Sound-Hardware holt sich ganz selbständig die Daten, die sie zur Tonerzeugung benötigt, aus dem Speicher des Amiga 500. Erst wenn diese Daten zu Ende sind, meldet sie dies dem Programm und bittet um neue Daten. Zwischendurch aber wird das eigentliche Programm nicht durch die Erzeugung von Musik beeinträchtigt oder gebremst.

Der Amiga 500 wird durch seine drei Zusatz-Chips ein echter Multi-Prozessor-Rechner, in dem zu jedem Zeitpunkt mehrere Mikroprozessoren gleichzeitig mit der Erfüllung der Wünsche des Benutzers beschäftigt sind. Er ist deshalb jedem Computer der gleichen Leistungsstufe, der alle Arbeit nur dem einen zentralen Mikroprozessor überläßt, haushoch überlegen!

1.7 Die Software des Amiga 500

Wie schon mehrfach erwähnt wurde, kann die Hardware eines Computers noch so beeindruckend und technisch hochentwickelt sein – ohne Software, die diese Hardware entsprechend nutzt, bleibt sie unwichtig und nutzlos. Sie werden nun ein wenig über diese Software erfahren. Dabei geht es aber noch nicht um die echten Anwendungsprogramme, die Ihnen zum Beispiel die Textverarbeitung oder Buchhaltung erleichtern. Sie erhalten statt dessen einen kurzen Überblick über die sogenannte System-Software, mit der Sie immer zu tun haben, wenn Sie einen Computer bedienen, ohne daß es Ihnen die meiste Zeit bewußt wird.

Diese System-Software erlaubt unter anderem auch erst die Kommunikation der Computer-Hardware mit der Außenwelt. (Diese Außenwelt sind Sie selbst, aber auch Drucker und andere Geräte, die Sie an Ihren Amiga 500 anschließen.) Sie stellt zum Beispiel auch die Vorgänge im Innern des Computers oder den Inhalt des Speichers am Bildschirm dar.

1.7.1 Die Workbench

Nach dem Einschaltvorgang, der im folgenden Kapitel noch in aller Ausführlichkeit erläutert wird, erscheint normalerweise die »Workbench«. Die Workbench (zu deutsch *Werkbank*) ist die »Oberfläche« (der Computer-Fachmann sagt auch *Shell*, zu deutsch *Schale*) des Amiga. Von dieser Werkbank aus können Sie Programme starten, Dateien und Disketten kopieren, löschen und umbenennen.

Die Disketten, Dateien und Programme werden als Symbole auf dem Bildschirm dargestellt. Alle wichtigen Operationen können mühelos durchgeführt werden, indem Sie einfach mit einem kleinen Zeiger, den man mit der Maus bewegen kann, auf bestimmte Objekte zeigen und einen der beiden Knöpfe auf der Maus drücken. Bestimmte Bewegungen dieses Zeigers zu bestimmten Punkten können ebenfalls Befehle bedeuten.

Ihre Dateien und Programme können Sie in Schubladen ablegen und so etwas Ordnung in Ihre Disketten bringen. Diese Schubladen werden Ihnen am Bildschirm auch wirklich als kleine Schubladen gezeigt. Sie können diese öffnen und nachschauen, welche Dateien darin enthalten sind. Sie können sie aber auch schließen und auf diese Weise Platz auf dem Bildschirm schaffen, weil Sie nicht mehr alle Dateien sehen, sondern nur noch die Schublade.

Das soll aber zunächst einmal an Informationen über die Workbench ausreichen. Sie wird noch in großer Ausführlichkeit in den folgenden Kapiteln des ersten Buchteils beschrieben.

Wie Sie aber recht bald bemerken werden, ist diese grafische Art der Computer-Bedienung sehr leicht zu erlernen und auch zu behalten. Sie brauchen sich nur sehr wenige Dinge zu merken. Die Möglichkeiten, die Ihnen in jedem Augenblick zur Verfügung stehen, werden Ihnen immer gezeigt. Kurz: Die Bedienung ist sehr intuitiv. (Das Paket von Programmroutinen in der Amiga-Systemsoftware, das für diese Dinge zuständig ist, heißt deshalb treffenderweise auch »Intuition«.)

1.7.2 Das CLI

Eine zweite, völlig verschiedene Methode, den Amiga 500 zu bedienen, stellt das CLI dar. »CLI« bedeutet *command line interface*, was man in etwa mit *Kommandozeilen-Schnittstelle* übersetzen könnte. Diese Bedienungsweise ist der herkömmlichen Bedienungsweise, wie sie auch auf anderen Computern üblich ist, sehr ähnlich. Sie müssen dabei einen Befehl, meist ein kurzes englisches Wort, und einige Parameter eintippen, die diesem Befehl sagen, was er genau tun soll. Die Ergebnisse des Befehls werden Ihnen dann üblicherweise auch in Textform am Bildschirm aufgelistet. Der Amiga wird beim CLI quasi als »alphanumerisches Terminal« mißbraucht.

Für viele Anwender gibt es wenig Gründe, sich mit dem CLI zu beschäftigen. Befehle in Textform sind wesentlich langwieriger einzugeben, schwerer zu behalten, und die teilweise langen Listen notwendiger Parameter ergeben viele Möglichkeiten, Fehler zu machen. Die Workbench ist eine viel »freundlichere« Umgebung. Einige Dinge gehen aber mit einer textuellen Bedienungsweise, also dem CLI, wesentlich einfacher. Die Programmierung zum Beispiel ist ein solches Anwendungsgebiet, in dem man mit dem CLI viel zügiger arbeiten kann. Das CLI als zusätzliche Möglichkeit, den Amiga zu bedienen, hat also durchaus seine Daseinsberechtigung.

Über das CLI werden Sie im zweiten Teil dieses Buches noch mehr erfahren; das Amiga-Handbuch schweigt sich darüber leider aus.

1.7.3 Multitasking

Eine der Software-Eigenschaften, die es vor dem Amiga bei Personalcomputern so gut wie gar nicht gab, ist die Fähigkeit, mehrere Programme gleichzeitig laufen zu lassen. Programme nennt man in diesem Zusammenhang oft auch Tasks oder Prozesse, und die Eigenschaft, mehrere Programme gleichzeitig verarbeiten zu können, heißt dementsprechend Multitasking.

Wenn man von der Workbench aus ein Programm startet, übernimmt es nicht den ganzen Computer, wie das bei Computern anderer Firmen üblich ist. Die anderen Programme, die zu diesem Zeitpunkt schon aktiv waren, zum Beispiel auch die Workbench selbst, laufen einfach weiter. Das hört sich im ersten Moment sehr beeindruckend an, insbesondere, weil dazu bis heute nur wesentlich größere und teurere Rechner fähig waren. Andererseits fragt man sich dann bald auch, wozu das alles. Schließlich kann man ja keine zwei oder drei Programme gleichzeitig bedienen, selbst wenn sie gleichzeitig laufen können.

Dieser Einwand stimmt, und man soll die Vorteile des Multitasking auch nicht überschätzen. Andererseits ist Multitasking bei langwierigen Vorgängen aber auch eine ganz praktische Sache. Die meisten Druckvorgänge laufen zum Beispiel auf dem Amiga standardmäßig »im Hintergrund«, also als separater Task. Während der Drucker rattert, kann man sich schon einmal anderen Dingen zuwenden. Genauso kann während des Formatierens (Vorbereitung zur Datenaufnahme) einer Diskette bereits mit anderen Programmen gearbeitet werden. Auch langwierige Sortier- oder Rechenvorgänge können sehr gut ohne Ihren andauernden Eingriff ablaufen, und Sie können unterdessen schon einmal einen Brief schreiben. Und schließlich ist

auch der Wechsel von einem Programm zum anderen, ein Vorgang, den Sie häufiger benötigen werden, als Sie jetzt vielleicht meinen, wesentlich schneller, wenn jedes Programm nicht immer neu gestartet werden muß.

Alles in allem sind die Multitasking-Möglichkeiten des Amiga sehr vielversprechend. Ich bin sicher, daß Sie diese Möglichkeit nicht mehr missen wollen, wenn Sie sie erst einmal eine Weile genutzt haben.

1.8 Etwas Historie

Zum Abschluß dieser Einleitung noch ein paar Informationen zur Geschichte des Amiga 500. Es ist ja immer ganz interessant, auch das Vorleben einer neuen Freundin (oder eines neuen Freundes) kennenzulernen.

1.8.1 Die Gerüchteküche kocht

Wie kein anderer Computer vor ihm, wurde der Amiga schon lange vor seinem wirklichen Erscheinen von den wildesten Gerüchten begleitet. Die ersten tauchten etwa im Winter '83 auf der COMDEX (eine große und wichtige Computermesse, die jedes Jahr einmal im Frühjahr und einmal im Herbst in den USA stattfindet) in den USA auf. »Amiga« war damals noch der Name einer Firma, die nur »Joysticks«, die kleinen Steuerknüppel für Videospiele, verkaufte – sehr gute allerdings. Einen Personalcomputer zu bauen und noch dazu einen, der alles bisher Dagewesene in den Schatten stellen sollte, traute dieser Firma eigentlich kaum jemand zu.

Als man jedoch den Namen eines der Männer hörte, die bei der Entwicklung des Amiga mitarbeiten sollten, wurden die Zweifel etwas geringer. Für das Design der drei Spezial-Chips und eines großen Teils der grundsätzlichen Architektur des Amiga ist Jay Miner verantwortlich, der auch schon die Grafik-Chips für die Atari 400/800er Serie entwarf, die auch heute noch im Atari 130 XE eingesetzt werden. Es sind im wesentlichen diese Chips, die für die relativ guten Grafikfähigkeiten der alten 8-Bit-Ataris (und für den Ruhm Miners) verantwortlich sind. Die Philosophien hinter dem Atari-Chip-Satz und den drei Custom-Chips des Amiga 500 sind sich sehr ähnlich, nur sind die Amiga-Chips ungleich leistungsfähiger, da sie dem neuesten Stand der Mikroprozessor-Technik entsprechen. Der Amiga kann deshalb mit Fug und Recht als ein Nachfolger der alten Ataris bezeichnet werden – eher jedenfalls als die neue Atari-ST-Serie, die eine völlige Neuentwicklung ist. Die Chips von Jay Miner sollten diesem neuen Heimcomputer Fähigkeiten geben, wie sie sonst nur in speziellen Grafikcomputern für einige Zehntausend Mark zu finden sind.

So waren denn auch alle Journalisten, denen Prototypen des neuen Amiga (oder »Lorraine«, wie er damals noch hieß) gezeigt wurden, hellauf begeistert. Die Firma Amiga machte in der Folge allerdings eine vor allem auch finanziell schwierige Zeit durch. Die Entwicklung der Grafik-Chips dauerte unerwartet lange und wurde dadurch natürlich auch unerwartet teuer. Es wurde dadurch mit der Zeit recht unsicher, ob der Amiga jemals das Licht der Computerwelt erblicken würde.

1.8.2 Finanzielle Turbulenzen

In der Zwischenzeit brach nämlich auch der Heimcomputer-Markt mehr oder weniger in sich zusammen. Mit neuen Heimcomputern glaubte niemand mehr Geld verdienen zu können. Der Amiga wurde (allerdings nicht allein deshalb) immer größer und teurer und mauserte sich langsam aber sicher zu einem vollständigen Personalcomputer, der nichts mehr mit den beschränkten Heimcomputern der damaligen Zeit zu tun hatte. Die Firma Amiga benötigte allerdings immer mehr Geld für die immer länger dauernde Entwicklung eines sich kontinuierlich ändernden Computer-Projekts. Amiga hatte auch Geld von der Firma Atari für die Entwicklung neuer Grafik-Chips bekommen. Als sich das Geschäft mit Atari zerschlagen hatte, mußten diese Summen zusammen mit hohen Vertragsstrafen zurückgezahlt werden. Kurz: Amiga war in großen Schwierigkeiten.

1.8.3 Der Amiga erblickt das Licht der Computerwelt

Glücklicherweise fand sich – wie wir alle inzwischen wissen – aber rechtzeitig ein finanzstarker Geldgeber: Commodore. Im Sommer '85 wurde der Amiga dann endlich im Rahmen einer millionenschweren Multimedia-Show in New York vorgestellt. (Seine deutsche Premiere hatte er auf der Hannovermesse/CeBIT im März '86.) Kaum einen der Anwesenden dürfte die Vorstellung kaltgelassen haben. Ganz abgesehen von reichlich Prominenz, mit der sich Amiga schmückte, so zum Beispiel dem inzwischen verstorbenen Pop-Künstler Andy Warhol und der Rocksängerin Debby Harrie (Blondie), wurden vor allem die grafischen und musikalischen Fähigkeiten des Amiga groß herausgestellt. Aus dem preiswerten Heimcomputer für etwa 500 Dollar, den Amiga zunächst entwickeln wollte, war inzwischen ein leistungsfähiger PC geworden, der auch in der Geschäftswelt Erfolg haben sollte (aber auch circa das Dreifache kostet).

In diesen fast zwei Jahren zwischen den ersten Gerüchten über den Amiga und seiner öffentlichen Präsentation kursierten die wildesten Gerüchte über eine wahre Supermaschine durch die Presse und die Computernetzwerke. Zu einem Zeitpunkt wurden die Gerüchte sogar Amiga selbst zu unglaublich, worauf ein Papier veröffentlicht wurde, in dem eine ungefähre Beschreibung an die Presse gegeben wurde, was die Maschine kann, aber auch, was sie nicht kann.

1.8.4 Startschwierigkeiten

Nach dem spektakulären Auftakt hatte der Amiga (genauer der Amiga 1000) allerdings einige Startschwierigkeiten. Das von Commodore angepeilte Publikum »Geschäftswelt« konnte aus verschiedenen Gründen heraus nicht so recht mit dem neuen Computer warmwerden. Einer der wichtigsten Gründe dürfte dabei die mangelnde Verfügbarkeit professioneller Software gewesen sein. Dieser Mangel dauerte über ein Jahr an und ist erst seit Anfang 1987 wirklich behoben.

Als Heimcomputer wäre der Amiga 1000 andererseits ganz hervorragend geeignet gewesen und hätte sicherlich auch großen Erfolg gehabt – wäre da nicht der hohe Preis gewesen. In Deutschland kostete der Amiga 1000 bei der Markteinführung zusammen mit einem

hochwertigen RGB-Monitor zum Beispiel fast 6000 DM. Diese Entwicklung bedeutete einmal mehr stürmische Zeiten für den Amiga – und Commodore.

Commodore geriet in der Folgezeit – nicht allein wegen des Amiga – in große finanzielle Schwierigkeiten, die die Zukunft des Amiga wieder ungewiß erscheinen ließen. Eine Zeitlang sah es fast so aus, als wäre das Ende des Amiga gekommen. Die Amiga-Entwicklungsabteilung wurde aufgelöst und sämtliche Weiterentwicklungen des Amiga-Konzepts gestoppt. Die ehemaligen Mitglieder der Firma Amiga, nun Angestellte von Commodore, wurden entlassen.

1.8.5 Die zweite Generation: Amiga 500 und Amiga 2000

Doch auch diese Schwierigkeiten haben Commodore und der Amiga inzwischen überwunden. Man kann sogar sagen, daß die Schwierigkeiten relativ unbeschadet überstanden wurden und die Amiga-Familie heute besser dasteht als jemals zuvor. An zwei verschiedenen Stellen in der Welt – im Commodore-Stammwerk in West Chester, USA, und in Deutschland im Commodore-Werk Braunschweig – wird die Amiga-Entwicklung fortgeführt. Nachdem die Preise gehörig gesenkt wurden, hat der Amiga inzwischen sowohl in den USA als auch hier bei uns in Europa große Verkaufserfolge zu verzeichnen. Im Weihnachtsgeschäft '86 ging der Amiga über die Ladentheken wie die sprichwörtlichen »warmen Semmeln«. Sowohl »Aufsteiger«, die zuvor einen weniger leistungsfähigen Computer besessen hatten, als auch Leute, die zuvor noch nie mit einem Computer gearbeitet hatten, entschieden sich für den Kauf eines Amiga.

Auch die Weiterentwicklung des Amiga hat inzwischen stattgefunden – in zwei völlig getrennten Richtungen. In Braunschweig entstand der Amiga 2000, eine leistungsfähige und ungemein ausbaufähige Weiterentwicklung des Amiga 1000. In West Chester, dem Commodore- Stammwerk in den USA, wurde gleichzeitig der Amiga 500 entwickelt.

Der Amiga 2000 besitzt von vorneherein schon 1 Mbyte RAM-Speicher und im Gehäuse Platz für insgesamt 3 Diskettenlaufwerke und eine Festplatte. In seinem relativ großen Gehäuse sind Anschlüsse (Steckplätze) vorhanden, in die man Erweiterungen (mehr Speicher, schnellere Prozessoren, andere Video-Adapter und so weiter) einsetzen kann. Mit einer speziellen Einsteckkarte kann man den Amiga 2000 auch MS-DOS-fähig und sogar IBM-kompatibel machen und so eine Vielzahl der sehr professionellen Softwarepakete nutzen, die für IBM-PCs entwickelt wurden. Einige der Steckplätze sind sogar fähig, Steckkarten, die für IBM-PCs entwickelt wurden, aufzunehmen. Der Amiga 2000 ist damit sogar hardwarekompatibel zu IBM-PCs. Für nicht einmal 5000 DM erhält man mit dem Amiga 2000 eigentlich zwei Computer auf dem Platz und zum Preis von einem.

Während der Schwerpunkt bei der Entwicklung des Amiga 2000 eindeutig bei der Steigerung der Leistungsfähigkeit und vor allem auch bei der Ausbaufähigkeit lag, wurde der Amiga 500 von vorneherein als kompakter, preiswerter Heimcomputer entwickelt. Er vereint Systemeinheit, Tastatur und ein Diskettenlaufwerk in einem kompakten, handlichen Gehäuse. Auch der interne Aufbau wurde vereinfacht und besser gegliedert, was vor allem durch neue, noch leistungsfähigere Chips möglich wurde, die Commodore in der Zwischenzeit entwickelt hat. Alles das hat zur Konsequenz, daß der Amiga 500 wesentlich kostengünstiger gebaut – und

deshalb auch wesentlich preiswerter verkauft – werden kann. Nicht nur äußerlich erinnert der Amiga 500 etwas an den C 64 – den erfolgreichsten Computer aller Zeiten – oder dessen Nachfolgemodell C 128. Der Amiga 500 ist in jeder Hinsicht der rechtmäßige Erbe des in die Jahre gekommenen C 64, der nicht mehr ganz dem Stand der Technik entspricht.

Man sollte sich aber von den Äußerlichkeiten dieses kleinen, fast unscheinbaren Computers nicht täuschen lassen. Der Amiga 500 ist ein echter Wolf im Schafspelz. In seinem kompakten Gehäuse enthält er alles (sogar etwas mehr), was in dem wesentlich größeren Amiga 1000 zu finden ist. Das Gehäuse ist das eines Heimcomputers, sein Inhalt und seine Leistungsfähigkeit entsprechen einem vollwertigen PC!

2 Erste Handgriffe auf der Werkbank

Das Programm, mit dem Sie einen Großteil Ihrer Arbeit am Amiga machen werden, ist die *Workbench* oder zu deutsch *Werkbank*. Von dieser Werkbank aus werden Sie Ihre Disketten verwalten, Dokumente anlegen, kopieren, umbenennen, ablegen, löschen, Programme starten und so weiter. Prinzipiell laufen fast alle Dinge, die nicht unmittelbar etwas mit einem sogenannten Anwendungsprogramm zu tun haben, auf der Workbench ab.

2.1 Los geht's: Starten des Amiga 500

Um auf die Workbench zu gelangen, müssen Sie natürlich zunächst einmal Ihren Amiga und den zugehörigen Monitor einschalten. Nachdem Sie dies getan haben, blinkt eine ganze Weile ein rotes Lämpchen am linken Rand der Haupteinheit. Der Amiga testet sich in dieser Zeit selbst. Er prüft währenddessen zum Beispiel seinen Speicher daraufhin, ob man aus diesem genau dieselben Zahlen lesen kann, die kurz zuvor hineingeschrieben wurden, schaut nach, ob die Speichererweiterung installiert ist und so weiter. Wenn alles zu seiner Zufriedenheit ausgefallen ist, bittet er Sie, die Workbench-Diskette einzulegen. Er zeigt Ihnen dazu das Bild einer Hand, die eine Diskette hält.

Eine kleine Sicherheitsvorkehrung sollten Sie dabei beachten: Bevor Sie eine der Disketten, die Ihrem neuen Amiga beim Kauf beilagen, in den Schlitz des Diskettenlaufwerks schieben, aktivieren Sie bitte deren Schreibschutz! Dies geht, indem Sie das kleine Plastikplättchen, das in einer der Diskettenecken zu finden ist, so verschieben, daß eine Öffnung entsteht, durch die Sie hindurchsehen können. Sie verhindern auf diese Weise, daß eine der relativ empfindlichen Disketten bei einem kleinen Fehler Ihrerseits, den Sie bei den folgenden Experimenten machen könnten, beschädigt werden kann. (Falls Ihnen diese kurze Erklärung des Schreibschutzes nicht genügt, können Sie eine etwas detailliertere am Ende dieses Kapitels finden.)

Bild 2.1 zeigt Ihnen, wie die Aufforderung, die Workbench-Diskette einzulegen, im wesentlichen aussieht. Achten Sie auf den Text, der – verkehrt herum – auf der Diskette steht.

Selbst wenn Sie stolzer Besitzer eines zweiten (externen) Diskettenlaufwerks sind, hilft es Ihnen nichts, die Workbench-Diskette in dieses zu schieben. Der Amiga sucht (und findet) die Workbench-Diskette nur im internen Laufwerk!

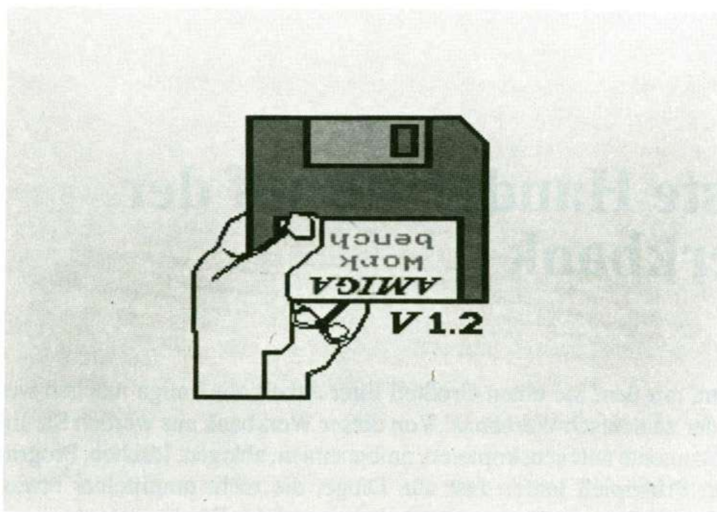


***Bild 2.1:** Die Aufforderung zum Einlegen der Workbench-Diskette*

Erscheint das Bild 2.1 erneut auf Ihrem Bildschirm, obwohl Sie die Workbench-Diskette eingelegt haben, so ist Ihre Workbench-Diskette wahrscheinlich defekt. Wenn Sie noch eine Kopie davon besitzen, versuchen Sie es mit dieser Kopie noch einmal. Falls das auch nichts fruchtet, schalten Sie Ihren Amiga noch einmal ganz aus und nach circa 10 Sekunden wieder ein. Klappt es auch jetzt nicht, bleibt Ihnen nichts anderes übrig, als sich an Ihren Amiga-Händler zu wenden.

Ist aber alles gut gegangen und der Amiga hat Ihre Workbench-Diskette »verdaut«, so verschwindet Bild 2.1 nach wenigen Sekunden, und das interne Diskettenlaufwerk beginnt hörbar zu arbeiten. Sie können übrigens immer an zwei Dingen feststellen, daß der Amiga auf die Disketten in einem der Diskettenlaufwerke »zugreift« (Daten davon liest oder darauf schreibt). Zum einen hören Sie dann ein leises, schabendes, schnarrendes oder surrendes Geräusch und zweitens leuchtet das kleine grüne Lämpchen oben auf dem Amiga (Diskettenlampe) auf. Während Sie eine dieser beiden Tatsachen registrieren, also die Diskettenlampe an ist oder Sie das Diskettengeräusch hören, sollten Sie niemals auf den Auswurfknopf unter dem Diskettenschlitz drücken! Diese Warnung werden Sie noch mehrmals lesen. Sie ist aber so wichtig, daß sie einige Wiederholungen verträgt. Gerade die 3,5-Zoll-Disketten des Amiga sind recht robust. Es bekommt aber keiner Diskette – und auch

keinem Laufwerk – wenn die Diskette aus dem Laufwerk entfernt wird, während der Computer auf ihre Daten zugreift.

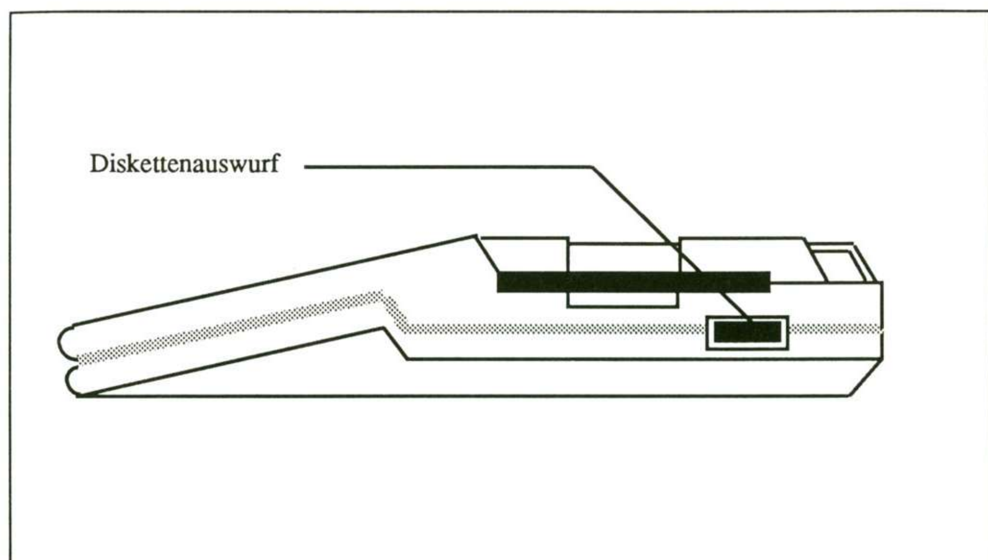


Bild 2.2: *Ein Amiga-Diskettenlaufwerk*

Nach einem kurzen Moment, währenddessen der Amiga Informationen von der Workbench-Diskette einliest, belebt sich der Bildschirm des Amiga wieder, und es erscheinen eine Copyright-Meldung, einige Meldungen, die Sie jetzt noch nicht verstehen werden, und dann schließlich die Workbench am Bildschirm. Der Bildschirm färbt sich blau und am oberen Bildschirmrand sehen Sie einen weißen Streifen, in dem am linken Rand steht: *Workbench release 1.2: 368248 free memory*. (Die Zahl vor *free memory* kann bei Ihnen allerdings eine andere sein.) Dieser Streifen heißt »Titelzeile« oder »Titelleiste« des Bildschirms.

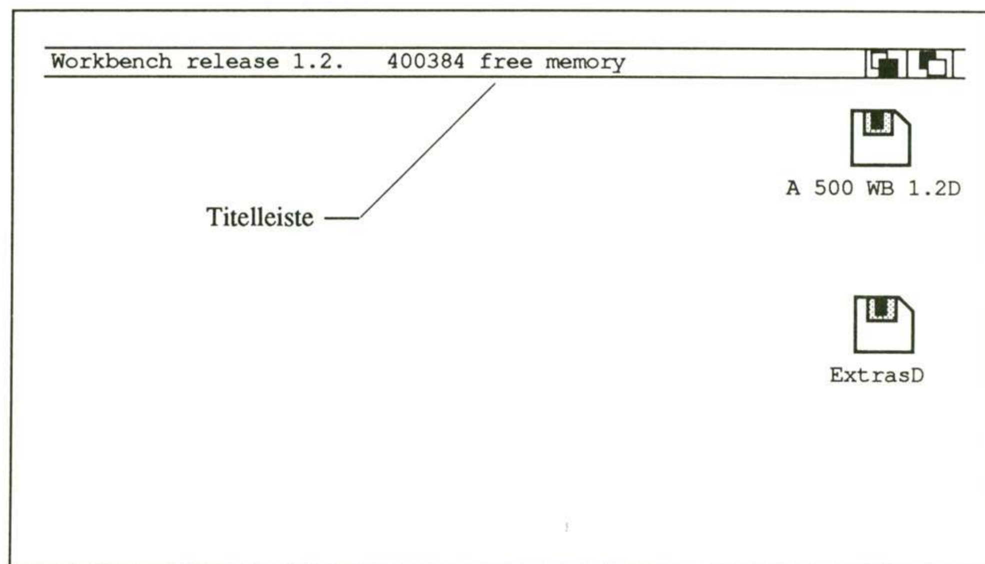


Bild 2.3: Die Titelleiste des Workbench-Bildschirms

Am rechten Rand der Titelzeile stehen zwei kleine Symbole, über die Sie in einem späteren Kapitel noch mehr erfahren werden. Falls Sie ein zweites Diskettenlaufwerk besitzen und sich in diesem schon eine Diskette befand, als Sie den Amiga gestartet haben, so erscheinen am rechten oberen Rand der blauen Fläche zwei kleine Bilder, die Disketten zumindest nicht unähnlich sind. Dies sind *Disketten-Piktogramme* oder *Disketten-Piktogramms*. (Die kleinen Bildchen, die beim Amiga die verschiedenen Objekte symbolisieren, mit denen wir es zu tun haben, nennen wir generell *Piktogramme* oder *Icons*.) Besitzen Sie nur ein Laufwerk oder hatten Sie keine Diskette in Ihrem zweiten Laufwerk, so sehen Sie dort nur eines dieser *Piktogramme*. Sie befinden sich nun auf der Workbench!

Wollen Sie die Workbench wieder verlassen, also den Amiga ausschalten, nehmen Sie bitte zuvor die Workbench-Diskette und/oder sämtliche anderen Disketten aus dem externen Diskettenlaufwerk. Ein Druck auf den Auswurfknopf an jedem Diskettenlaufwerk macht Ihren Disketten in diesem Fall das Leben leichter. Sie sollten niemals »radikale« Aktionen, wie das Ein- oder Ausschalten des Amiga, durchführen, solange sich eine Diskette in einem Laufwerk befindet!

Bevor wir nun in die Tiefen der Workbench eindringen, noch ein Wort zur Terminologie. Der Vorgang, den der Amiga soeben durchlaufen hat, war der Einschaltvorgang. Er beinhaltet das Einschalten des Amiga und das Starten der Workbench mit Hilfe der Workbench-Diskette.

Das Starten der Workbench heißt auch *Neustart* des Amiga. Der Computerfachmann sagt dazu auch oft *Rebooten*. Selbst nach einem schwerwiegenden Programmfehler, der Ihren Amiga

total »in Unordnung« gebracht hat, genügt immer ein Neustart, um wieder »sauber« von vorne anzufangen.

Der Begriff *booten* oder *rebooten* kommt übrigens vom englischen Wort *bootstraps* (Schnürsenkel) und hat etwas mit der Vorstellung zu tun, daß man sich selbst an seinen Schnürsenkeln aus einem Sumpf zieht. Im deutschsprachigen Raum bevorzugt man allerdings das Ziehen an den eigenen Haaren – siehe Münchhausen. Wenn man versteht, was beim Start eines Computers mittels Software, die er erst von der Diskette lesen muß, vor sich geht, liegt dieser absurde Vergleich gar nicht einmal so fern.

2.2 Wie Sie einen Neustart erzwingen

Sollte Ihnen einmal das schreckliche Ereignis eines solchen Programmabsturzes widerfahren oder sollte es Ihnen passieren, daß Ihr Amiga einmal auf keinen Ihrer Versuche, seine Aufmerksamkeit zu erregen, reagiert, dann können Sie einen Neustart erzwingen. Drücken Sie dazu die Taste mit dem [*C*=]-Symbol links und die Taste mit dem [*A*]-Symbol rechts neben der Leer-Taste auf der Tastatur und dann gleichzeitig auf die [Ctrl]-Taste. Sie können die Tasten auch nacheinander drücken; es muß nur einen kurzen Augenblick geben, in dem alle drei zugleich niedergedrückt sind. (Sie benötigen dazu beide Hände.) Bevor Sie dies machen, vergewissern Sie sich, daß keines der Diskettenlaufwerke mehr aktiv ist und nehmen Sie am besten auch alle Disketten aus den Laufwerken (durch Drücken auf die entsprechenden Auswurftasten).

Nachdem Sie die drei Tasten losgelassen haben, blinkt wieder die Betriebsleuchte links auf dem Gehäuse des Amiga auf und nach kurzer Zeit erscheint die Aufforderung, die Workbench-Diskette einzulegen (Bild 2.1). Verfahren Sie dann wie oben beschrieben.

Die Tastenkombination für den Neustart sollte nur in absoluten Notfällen angewandt werden! Alle Daten, die zu dem Zeitpunkt, zu dem Sie den Amiga neu starten, noch nicht auf Disketten gesichert waren, gehen dabei verloren. Aus Versehen kann diese Tastenkombination glücklicherweise aber kaum vorkommen, außer Sie besitzen 20 Zentimeter lange Finger oder lassen Gegenstände auf die Tastatur fallen und halten dabei gleichzeitig Tasten fest.

Wenn Sie gerade auf der Workbench sind und keine Programme mehr aktiv sind, probieren Sie den »gewaltsamen« Neustart doch einfach einmal aus! Warten Sie dazu, bis sämtliche Disketten-Aktivitäten beendet und alle entsprechenden Lampen erloschen sind und nehmen Sie dann die Diskette(n) aus dem (den) Laufwerk(en). Dann drücken Sie die drei Tasten nieder und lassen sie wieder los.

2.3 Wozu die Workbench-Diskette gut ist

Bevor wir uns näher mit der Bedienung des Amiga beschäftigen, nun noch ein Wort zur Workbench-Diskette. Besonders, wenn Sie schon Kontakt mit anderen Computern, insbesondere Heimcomputern, hatten, werden Sie sich vielleicht wundern, wozu diese Diskette gut ist. Bei vielen anderen Computern reicht es ja, sie einfach einzuschalten und man kann mit der Arbeit beginnen. Beim Amiga ist das etwas anders.

Jeder Computer benötigt, damit Sie mit ihm arbeiten können, ja die sogenannte *Software*, also Programme. Erst ein Textverarbeitungsprogramm »bringt ihm bei«, Ihre Eingaben auf dem Bildschirm darzustellen und gegebenenfalls abzuspeichern. Der Computer selbst, die Hardware, ist ohne Software unfähig. Einen Teil dieser Software werden Sie allerdings niemals bemerken. Diese sogenannte *Systemsoftware* ist unsichtbar im Hintergrund tätig und erleichtert den Anwendungsprogrammen, wie der eben schon erwähnten Textverarbeitung, die Arbeit. Die Systemsoftware verwaltet zum Beispiel den Speicher des Computers und teilt ihn den verschiedenen Programmen, die Speicher benötigen, zu. Sie legt auf Wunsch der Anwendungsprogramme Daten auf der Diskette oder Festplatte ab und findet sie später auch wieder auf. Und schließlich sorgt sie zum Beispiel auch dafür, daß auf einen einfachen Programmbefehl, wie *Linie von (10,10) zu (100,100),* hin tatsächlich eine Linie auf dem Bildschirm erscheint.

Bei vielen preiswerten Heimcomputern ist die Systemsoftware in Form sogenannter ROMs vollständig in den Computer eingebaut. ROMs sind spezielle Speicherchips, die Daten enthalten, die nur gelesen und nie wieder geändert werden können (ROM = *Read Only Memory*; oder auf deutsch *Nur-Lese-Speicher*). Dies hat den Vorteil, daß sie jederzeit – und vor allem auch sofort nach dem Einschalten – zur Verfügung steht. Ein Nachteil dieser Lösung ist allerdings, daß die Software sehr schlecht geändert werden kann, wenn nachträglich Fehler darin gefunden werden oder wenn Verbesserungen eingeführt werden sollen. Beim Amiga hat man sich deshalb für eine zweigeteilte Lösung entschieden: Ein Teil der Systemsoftware ist fest in ROMs eingebaut und ein anderer Teil befindet sich auf der Workbench-Diskette und wird beim Start eingelesen. Dieser zweite Teil kann sehr leicht geändert und verbessert werden. Trotzdem wurde aus »Bequemlichkeit« nicht alle Systemsoftware auf diese Diskette verlagert. Die Software in den ROMs (256Kbyte) braucht nämlich beim Start nicht von der Diskette eingelesen zu werden, was den Startvorgang beschleunigt. Oder anders gesagt: Sie brauchen nach dem Einschalten nicht so lange zu warten, bis Sie mit der Arbeit beginnen können!

2.4 Ein besonderes Nagetier: die Maus

Das wichtigste Bedienungselement Ihres Amiga ist die Maus. Sie kann für einen großen Teil aller anfallenden Arbeiten auf der Workbench und auch innerhalb der meisten Programme, die Sie für den Amiga kaufen können, eingesetzt werden. Gerade auf der Workbench ist die Tastatur nahezu überflüssig – ohne Maus sind Sie hingegen (fast) rettungslos verloren. Dies wird Sie wahrscheinlich verblüffen, wenn Sie schon andere Computer kennen. Bei vielen anderen Computern (mit wenigen Ausnahmen) ist die Tastatur nämlich elementar für die ganze

Bedienung des Gerätes, und ohne Tastatur wären Sie völlig aufgeschmissen. Auf dem Amiga ist das anders – und deshalb einfacher, wie Sie bald merken werden. Die Maus – vor allem mit der zugehörigen Software – hat einige Vorteile gegenüber der üblichen Computer-Bedienung mit Befehlen, die über die Tastatur eingegeben werden müssen.

Dies soll nicht heißen, daß Sie auf dem Amiga Textverarbeitung ohne Tastatur betreiben können. Für viele Anwendungen ist die Tastatur immer noch unentbehrlich. Wenn man aber von den Vorgängen auf der Workbench zum normalen (nicht computerisierten) Leben eine Analogie zieht, dann kann man sagen, daß Sie immer dann ohne Tastatur auskommen werden, wenn Sie das beim entsprechenden Vorgang ohne Computer auch würden. Die Tastatur wird beim Amiga wirklich nur für die Texteingabe benutzt, und sonst nicht.

2.4.1 Funktionsweise der Maus

Bild 2.4 zeigt Ihnen die Maus des Amiga von oben. Die Ähnlichkeit mit einer lebenden Maus ist nicht gerade frappierend, aber mit etwas Mühe zu erkennen. Ich habe allerdings schon Computer-Mäuse gesehen, die von ihrem Besitzer einen Pelz verpaßt bekommen haben und dann doch recht putzig aussahen. Wenn Sie also ihren Computer gern warm und flauschig bedienen wollen... (geschickt gemacht, schadet eine solche Pelzhülle der korrekten Funktion überhaupt nicht).

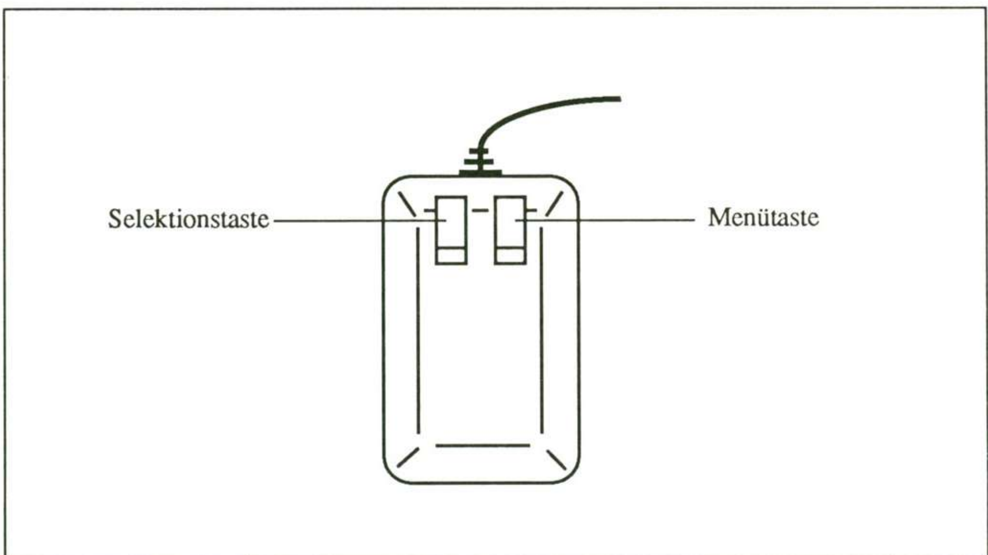


Bild 2.4: *Die Amiga-Maus*

Die Oberseite der Maus ist glatt und sollte gut in die locker darauf gelegte Handfläche passen. Ob sie das auch tut, darüber kann man geteilter Meinung sein – Geschmackssache. (Ihre Form

ist aber wohl unbestreitbar schnittig und modern.) An der Oberseite trägt sie zwei Tasten. Wird eine dieser Tasten gedrückt, so bewirkt dies bei den meisten Programmen eine bestimmte Aktion – abhängig vom gerade laufenden Programm können es allerdings verschiedene Aktionen sein.

Die linke der beiden Tasten heißt *Aktions-* oder *Selektionstaste*, die rechte *Menütaste*. (Keine Angst, Sie haben sich nicht in ein Kochbuch verirrt. Die Namen der Knöpfe werden weiter unten noch erläutert.)

Die Unterseite der Maus ist (zumindest für ihre Funktion) noch etwas interessanter. Falls Sie dieses Buch nicht gekauft haben, weil Sie erst noch Besitzer eines Amiga werden wollen, sondern es schon sind, nehmen Sie einmal die Maus zu Hand und drehen Sie sie auf ihre Rückseite. In der Mitte des Mausbodens sehen Sie ein kleines Loch und darin eine kleine Gummikugel. Immer wenn Sie die Maus – richtig herum – auf eine glatte Oberfläche legen und sie dann bewegen, dreht sich diese Kugel und überträgt diese Drehung auf zwei kleine Rädchen in ihrem Innern. Diese Rädchen senden bei Drehung elektrische Impulse zum Amiga, die ihm genau sagen, in welche Richtung und wie weit Sie die Maus bewegt haben.

Wie diese Impulse genutzt werden, ist prinzipiell natürlich Sache des gerade laufenden Programms. Meist werden sie aber dazu verwendet, einen kleinen Zeiger auf dem Bildschirm synchron zur Maus zu bewegen.

2.4.2 Erste Maus-Spiele

Falls Sie noch nicht mit der Maus vertraut sind, sollten Sie jetzt versuchen, es zu werden. Begeben Sie sich dazu auf die Workbench (wie oben beschrieben) und nehmen Sie die Maus in die Hand. Legen Sie sie dann auf eine glatte Unterlage, die wenigstens die Größe eines DIN-A4-Blattes haben sollte. Später werden Sie lernen, auch mit weniger Platz auszukommen; zunächst kann es aber gar nicht genug Platz für sie geben. In welche Hand Sie die Maus nehmen, ist im wesentlichen Geschmacksfrage. Zu Beginn ist es aber wahrscheinlich einfacher, wenn Sie sie als Linkshänder in die linke und als Rechtshänder in die rechte Hand nehmen. Mit ein wenig Übung wird diese Unterscheidung aber bald gegenstandslos werden und Sie werden die Maus mit der Hand bewegen, in deren Nähe der meiste Platz auf dem Schreibtisch ist.

Wenn Sie die Maus nun auf dieser Oberfläche bewegen, können Sie sehen, wie ein kleiner roter Pfeil auf dem Bildschirm diesen Bewegungen folgt. Dies ist der *Mauszeiger* – das wichtigste Objekt auf dem Bildschirm. Die folgende Abbildung zeigt Ihnen diesen Zeiger in vergrößerter Form – leider nur in Schwarzweiß. Auf dem Bildschirm erscheint er vor allem rot, besitzt in Wirklichkeit aber mehrere Farbtöne, damit er auch über den unterschiedlichsten Hintergrundfarben sichtbar bleibt.

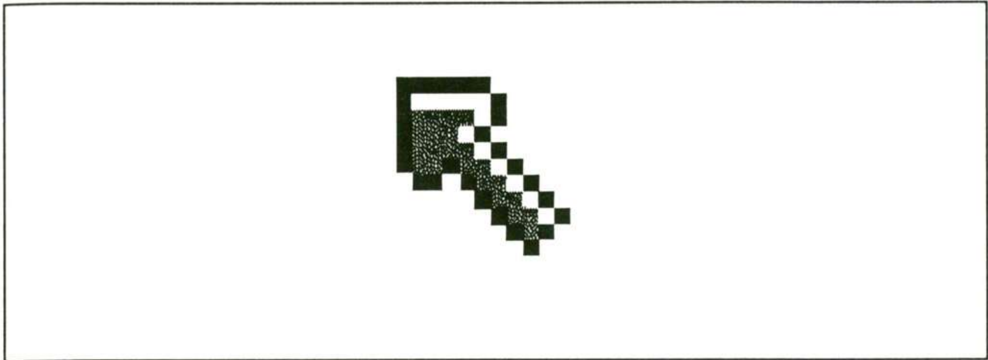


Bild 2.5: Der Mauszeiger des Amiga

Versuchen Sie nun einmal, die vier Ecken des Bildschirms mit diesem Zeiger gezielt anzufahren. Dies ist unter Umständen gar nicht so einfach, wenn Sie die Maus nicht richtig halten. Richtig gehalten wird die Maus immer so, daß ihr Schwanz (der Kabelanschluß) geradlinig auf den Bildschirm weist. Schieben Sie die Maus auf der Unterlage nach vorn, so geht der Zeiger auf dem Bildschirm nach oben. Ziehen Sie sie an Ihren Körper, geht der Zeiger nach unten. Bewegen Sie die Maus nach rechts, und der Zeiger geht nach rechts. Bewegen Sie sie nach links, und der Zeiger folgt nach links.

Drehen Sie die Maus testweise einmal so, daß der Schwanz nach links zeigt, und bewegen Sie sie dann nacheinander in die vier Bildschirmecken. Wie Sie sehen, ist dies viel schwieriger als eben mit dem Schwanz nach vorn, da die Entsprechung von Hand- beziehungsweise Mausbewegung zu Zeigerbewegung nicht mehr stimmt!

Versuchen Sie auch einmal, die Spitze des Mauszeigers genau auf die obere linke Ecke des Diskettensymbols in der rechten oberen Bildschirmecke zu bringen. Sie werden merken, daß es gar nicht so einfach ist, die Handbewegungen mit der Bildschirmdarstellung zu koordinieren und der Versuchung zu widerstehen, permanent auf die Maus zu schauen. Sie werden sich aber bald daran gewöhnen. Die Erfahrung zeigt, daß die Maus nach etwa ein bis zwei Stunden keinem Anwender mehr Probleme macht. Die Kopplung zwischen Maus und Mauszeiger wird auch Ihnen bald so natürlich erscheinen wie ein direktes Zeigen mit dem Finger. Sie ist sogar so selbstverständlich, daß in diesem Buch manchmal der Fehler gemacht wird, von der Maus zu sprechen, wenn eigentlich der Mauszeiger gemeint ist. Da die beiden unzertrennlich sind, ist dieser Fehler hoffentlich verzeihlich.

2.4.3 Die Tasten der Maus

Wie die Maus bewegt wird und was diese Bewegung bewirkt, haben wir ja jetzt gesehen. Auf der Oberseite der Maus befinden sich aber auch noch zwei Tasten. Da beide Tasten häufig gebraucht werden, sollten Sie sich eine Handhaltung angewöhnen, mit der Sie beide unabhängig voneinander niederdrücken können. Zeige- und Mittelfinger eignen sich dazu gut.

Sie können die Maus dann mit Daumen und Ringfinger sehr gut umfassen und diese beiden Finger immer locker auf den Knöpfen liegen lassen. Ganz so locker wird Ihre Handhaltung am Anfang zwar bestimmt nicht sein, aber keine Angst: das legt sich mit der Zeit (und etwas Übung).

Bewegen Sie den Mauszeiger nun einmal in die freie blaue Fläche in der Bildschirmmitte, drücken Sie dann kurz auf den linken Mausknopf und lassen ihn sofort wieder los. Es passiert nichts! Wenn Sie dabei aber auf den Bildschirm schauen, werden Sie feststellen, daß der Amiga Ihren Knopfdruck bemerkt hat. Die Titelzeile des Bildschirms flackert jedesmal leicht, wenn Sie den linken Knopf drücken. Dieser Vorgang des Drückens und Loslassens der Maustaste heißt *Mausklick* und das entsprechende Verb heißt *klicken* (mit der Maus). Merken Sie sich bitte diese beiden »Vokabeln«; Sie werden sie immer wieder benötigen.

Gehen Sie nun mit dem Mauszeiger über das Disketten-Piktogramm in der rechten oberen Bildschirmecke. Die Spitze des roten Pfeils sollte sich über der weißen Fläche des Symbols befinden. Drücken Sie nun wieder kurz die linke Maustaste und lassen sie sofort wieder los. Das Piktogramm wechselt daraufhin seine Farbe. Im folgenden heißt dieser Vorgang immer *Selektieren* oder *Auswählen* eines Objekts. Die linke Taste heißt dementsprechend ja auch Selektions-Taste oder Aktions-Taste.

Bewegen Sie den Zeiger nun wieder auf die blaue Bildschirmfläche und klicken Sie erneut mit dem linken Knopf. Das Diskettensymbol erhält daraufhin wieder seine alten Farben. Sie haben das Symbol *deselektiert*. Versuchen Sie es zur Übung gleich noch einmal: Anklicken des Disketten-Piktogramms (Selektieren) und dann Anklicken der blauen Fläche (Deselektieren).

Sie haben damit schon eine wesentliche Operation zur Bedienung der Workbench und des Amiga erlernt: das Auswählen eines Objektes zur Bearbeitung. Wenn Sie später irgendeinen Befehl geben, zum Beispiel *kopiere*, müssen Sie dem Amiga zuvor immer sagen, mit welchem Objekt er diesen Befehl durchführen soll (zum Beispiel was er kopieren soll). Das vorherige Anklicken eines oder mehrerer Objekte erfüllt genau diesen Zweck. Sie suchen sich zum Beispiel eine Diskette aus und sagen dem Amiga dann *kopiere* (diese Diskette).

Mit dem linken Knopf können Sie aber nicht nur Objekte auswählen und wieder deselektieren, sondern zum Beispiel auch Objekte bewegen. Dazu drücken Sie die linke Maustaste und halten sie dann gedrückt. Probieren Sie es aus: Bewegen Sie den Mauszeiger über das Diskettensymbol, drücken Sie die linke Taste und halten Sie sie gedrückt. Sie sehen, wie sich die Farben des Diskettensymbols ändern, sobald die Taste niedergedrückt wird. Bewegen Sie nun diesen Zeiger in die Mitte des Bildschirms und halten dabei die Maustaste immer unten. Das Diskettensymbol folgt dieser Bewegung.

Dieser Vorgang heißt *Ziehen der Maus*. Beim Ziehen bleibt im Gegensatz zum einfachen Bewegen die Maustaste gedrückt, während die Maus beziehungsweise der Mauszeiger bewegt wird. Sobald Sie die Maustaste nun loslassen, erscheint das Diskettensymbol an der Stelle, an der sich auch der Mauszeiger in diesem Moment befand. Sie haben das Disketten-Piktogramm auf diese Weise verlegt. Versuchen Sie, es nun wieder zurückzulegen. Dazu »ergreifen« Sie das Piktogramm wieder mit der Maus (drücken die Maustaste nieder, während der Zeiger sich

über dem Piktogramm befindet) und bewegen den roten Mauszeiger in die rechte obere Bildschirmecke. Dort lassen Sie die Maustaste los, und die Diskette liegt wieder (fast) an ihrer alten Stelle.

Wir werden später noch eine Reihe anderer Piktogramme kennenlernen, die auf der Workbench des Amiga erscheinen können. Alle diese Piktogramme haben gemeinsam, daß man sie mit der linken Maustaste zur Bearbeitung auswählen (selektieren) und sie bei gedrückter linker Maustaste auch ergreifen und bewegen kann.

Die Maus besitzt aber noch eine zweite Taste. Diese rechte Taste, die sogenannte »Menü-Taste«, haben wir überhaupt noch nicht verwendet. Falls Sie sie schon aus Versehen gedrückt haben, werden Sie gesehen haben, daß sie nicht dieselbe Funktion wie die linke hat – die obigen Beispiele sind »mit rechts« alle nicht durchführbar. Vielleicht haben Sie aber schon eine Veränderung auf dem Bildschirm bemerkt, als Sie die Menütaste herunterdrückten. Falls nicht, tun Sie es jetzt noch einmal.

Sobald Sie mit der rechten Taste klicken (und solange Sie sie festhalten), ändert sich die Titelzeile des Bildschirms. Anstelle des Textes *Workbench release 1.2......* erscheinen darin die drei Worte *Workbench*, *Disk* und *Special*. Dies sind die Titel der drei Menüs, aus denen Sie die Befehle der Workbench auswählen können. Solange die Menütitel sichtbar sind, nennen wir die Titelzeile des Bildschirms manchmal übrigens auch *Menüzeile*, um ihre neue Funktion dadurch noch etwas zu betonen. Menüs sind eine sehr bequeme Möglichkeit, einem Computer Befehle zu erteilen und zudem eine, die Eingabefehler fast unmöglich macht. In den nächsten Abschnitten werden Sie noch mehr darüber erfahren. Fast alle Amiga-Programme verwenden diese Möglichkeit. Natürlich gibt es in der Workbench mehr als drei Befehle. Die Befehle der Workbench sind aber in drei Gruppen oder auf drei Menüs aufgeteilt. Die Titel dieser drei Gruppen von Befehlen – und nicht die Befehle selbst – sind es, die in der Titelzeile erscheinen, wenn die Menütaste gedrückt wird.

2.5 Was darf es sein: Menüs

Mit diesen Menüs wollen wir uns nun etwas näher beschäftigen. Sie haben übrigens ihren Namen daher, daß sie dem Benutzer des Computers – also Ihnen – die Befehle anbieten, wie eine Speisekarte die Gänge eines Menüs zur Wahl anbietet. Guten Appetit also zu dieser Mahlzeit!

2.5.1 Wie Sie die Befehls-Menüs sichtbar machen

Um die Menüs des Amiga näher kennenzulernen, halten Sie zunächst einmal wieder die Menütaste (die rechte) der Maus nieder und bewegen Sie dann den Mauszeiger in die Titelzeile über das erste Wort *Workbench*. Das Bild, das erscheint, sobald der Zeiger sich über dem Wort *Workbench* befindet, dürfte ungefähr so aussehen wie die folgende Abbildung.

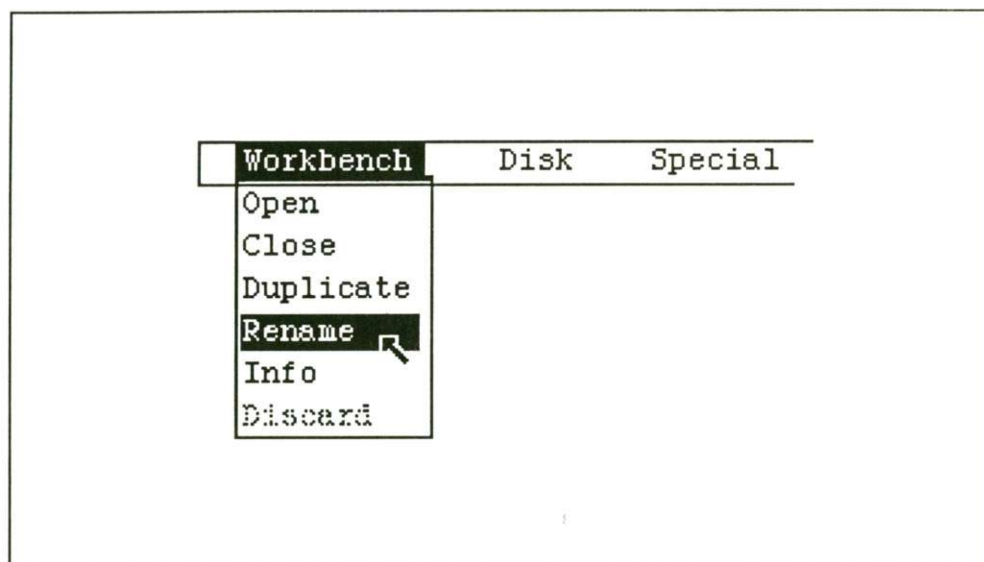


Bild 2.6: *Das Workbench-Menü*

Ein weißes Rechteck »klappt« herunter und zeigt Ihnen einige der Befehle, die die Workbench anbietet. Es kann allerdings durchaus sein, daß alle Befehls-Namen, die Sie sehen, so schlecht lesbar sind wie im obigen Bild das Wort *Discard*. (Die Gründe dafür werden Sie später noch erfahren. Zunächst soll uns das einmal nicht zu sehr kümmern.)

Bewegen Sie den Mauszeiger (bei gedrückter Menütaste) nun nach rechts über das nächste Wort *Disk*. Belassen Sie den Mauszeiger unbedingt in der Titelzeile und halten Sie die Maustaste gedrückt! (Falls Ihr Zeigefinger erlahmen sollte, bewegen Sie den Zeiger über die freie blaue Fläche, lassen dann die Menütaste los und beginnen nach einer Erholungszeit wieder von vorne.) Sobald sich der Mauszeiger über *Disk* befindet, verschwindet das erste Menü und ein neues taucht auf. Bewegen Sie ihn weiter nach rechts und der Vorgang wiederholt sich: das zweite Menü verschwindet und ein drittes erscheint unter dem Wort *Special*. Bewegen Sie die Maus in der Titelzeile hin und her und Sie können alle Befehle, die Ihnen in einem Programm – in diesem Fall dem Workbench-Programm – zur Verfügung stehen, schnell überblicken.

Auch für die Zukunft ist es eine gute Angewohnheit, bei gedrückter Menütaste über die Titelzeile des Bildschirms zu fahren, wenn man ein neues Programm erlernen muß oder einmal nicht mehr recht weiter weiß. Dies ist eine unschätzbare Orientierungshilfe in jedem Programm. Das Schöne an den Menübefehlen des Amiga ist auch, daß man jederzeit sieht, welche Befehle gerade sinnvoll möglich sind und welche nicht. Die in einem Moment nicht möglichen oder sinnlosen Befehle erscheinen stets blaß oder »geisterhaft«, während die Befehle, die sinnvoll sind und die Sie wirklich auswählen können, fett und gut lesbar sind.

Solange Sie keine Objekte in der Workbench ausgewählt haben, sind allerdings fast alle Einträge in den drei Menüs grau.

2.5.2 Wie Sie einen Menübefehl erteilen

Bis jetzt haben wir uns ja nur den Inhalt der Menüs angeschaut, jedoch noch nichts damit gemacht. Nun aber geht es »zur Sache«. In jedem Menü erscheint eine Reihe von kurzen Worten. Dies sind die Namen der einzelnen Menübefehle, die die Workbench versteht. Allgemein nennt man diese Worte manchmal auch Menüpunkte. Klappen Sie nun einmal das Menü *Special* herunter (wie das geht, haben Sie ja bereits im letzten Abschnitt erfahren) und belassen Sie den Mauszeiger über dem Namen *Special*.

Ihnen stehen jetzt die Befehle *Clean Up* (engl. für *aufräumen*), *Last Error* (engl. für *letzter Fehler*), *Redraw* (engl. für *neu zeichnen*) und Version (engl. für *Versionsnummer zeigen*) zur Verfügung. Einige davon sind wahrscheinlich grau. Niemals grau wird der Befehl *Version*, der uns die Versionsnummer der Software zeigt, und genau diesen wollen wir jetzt aufrufen.

Bei gedrückter Menütaste wird dazu die Maus innerhalb des weißen Kastens, der die Menüpunkte enthält, nach unten bewegt. Jeder Menüpunkt, der unter den Mauszeiger gerät und nicht grau gezeichnet war, wechselt dabei seine Farbe. Er erscheint orange auf schwarz statt blau auf weiß gezeichnet. Probieren Sie es einmal aus, indem Sie den Mauszeiger im Menü *Spezial* auf und ab bewegen. Lassen Sie dabei die Menütaste aber bitte nicht los! Sobald der Zeiger das Menü verläßt, werden alle Befehle wieder normal gezeichnet; das Menü bleibt aber sichtbar, solange die Menütaste gedrückt bleibt.

Bewegen Sie nun den Zeiger über das Wort *Version*, so daß es orange auf schwarz geschrieben wird. Lassen Sie jetzt die Menütaste los. Und nun stellen Sie sich bitte einen kleinen Tusch vor, denn soeben haben Sie dem Amiga den ersten Befehl erteilt! Er führt ihn auch sofort aus. Der Bildschirm flackert kurz auf und zeigt Ihnen dann oben links in der Titelzeile die genaue Versionsnummer der Workbench (zum Beispiel *Kickstart version 33.180 Workbench Version 31.47*). Zur Eingewöhnung versuchen Sie es am besten gleich noch einmal.

Jedesmal, wenn ein Menü-Punkt orange auf schwarz hervorgehoben wird und Sie in diesem Augenblick die Menütaste loslassen, erteilen Sie dem Amiga einen Befehl. Sie brauchen diesen Befehl also nicht wie bei anderen Computern einzutippen, sondern wählen ihn einfach aus dem Menü aus – wie Sie auch ein Gericht aus einem Restaurant-Menü auswählen würden.

Der *Version*-Befehl ist ein Menübefehl, der keine zusätzlichen Informationen über die Objekte, mit denen er arbeiten soll, benötigt. Dies stellt unter den Menübefehlen aber eher eine Ausnahme dar. Fast alle anderen Befehle benötigen solche zuvor ausgewählten Objekte, die der Computerfachmann *Argumente* beziehungsweise *Parameter* des Befehls nennt. Wir wollen uns nun einmal einen solchen Befehl mit Argumenten anschauen.

Selektieren Sie dazu das einzige Objekt, das bisher auf der Werkbank zu sehen ist: das Disketten-Piktogramm oben rechts am Bildschirm (durch einen Klick mit der linken Taste auf dieses Piktogramm). Dann drücken Sie wieder die Menütaste und gehen nun über das Wort *Workbench* in der Menüzeile. Der erste Befehl in diesem Menü heißt *Open* (engl. für *öffnen*).

Bewegen Sie die Maus über das Wort *Open* und lassen Sie die Taste los. Sie haben dem Amiga dadurch sinngemäß den Befehl *Öffne diese Diskette* gegeben.

Das erste, was geschieht, ist, daß ein weißer Rahmen auf dem Bildschirm erscheint. Dieser Rahmen trägt am oberen linken Rand den Titel *Workbench* und es tauchen nun langsam unter hörbarem Arbeiten des Diskettenlaufwerks nacheinander eine Reihe von Piktogrammen darin auf: einige Schubladen, eine Uhr, ein Mülleimer und ein Piktogramm, das dem Amiga selbst verdächtig ähnelt. Dieser weiße Rahmen und die Fläche in seinem Innern ist ein sogenanntes »Fenster«, in dem uns der Inhalt der Diskette namens Workbench gezeigt wird.

Diese Eigenschaft des *Open*-Befehls, die hier demonstriert wurde, sollten Sie sich unbedingt merken. Obwohl Sie den *Open*-Befehl auf die verschiedensten Objekte anwenden können, zeigt er Ihnen stets (meist in einem Fenster), was sich in diesem Objekt verbirgt. Das Pendant zum Befehl *Open* (öffnen) ist natürlich der Befehl *Close* (engl. für *schließen*). Dieser Befehl (*Close*) findet sich im Menü *Workbench* direkt unter *Open*. Wählen Sie ihn nun aus. Wie Sie sehen, verschwindet das Fenster daraufhin sofort und Sie blicken wieder auf die blaue Fläche der Workbench.

Wenn Sie den Befehl *Close* nicht auswählen können, weil er blaß erscheint, haben Sie, nachdem Sie *Open* gewählt haben, wahrscheinlich zwischendurch einmal auf die Selektionstaste gedrückt. Bewegen Sie dann den Mauszeiger über das Disketten-Piktogramm und drücken Sie einmal auf die Selektionstaste (Sie haben die Diskette dadurch natürlich selektiert). Nun sollten Sie *Close* auswählen können.

Klicken Sie nun wieder das Disketten-Piktogramm an und wählen Sie erneut *Open* aus. Beachten Sie diesmal auch, was dabei mit dem Mauszeiger geschieht. Sie werden bemerken, daß er seine Form ändert, während sich das Fenster, das aus dem Disketten-Piktogramm hervorgegangen ist, langsam mit Objekten füllt. Er wird zu einer Art Sprechblase (wie in einem Comic-Heftchen), in der zwei »Z« stehen. Der Amiga sagt Ihnen auf diese Weise, daß er beschäftigt ist (beziehungsweise ein kleines Nickerchen macht) und nicht mehr sofort auf alle Ihre Eingaben reagieren kann. Immer, wenn Sie diese kleine Sprechblase sehen, ist ein Programm mit einer etwas länger andauernden Tätigkeit beschäftigt. Falls Sie zur Ungeduld neigen, wird Sie diese Sprechblase bestimmt bald etwas nerven. Manche Dinge dauern nun aber einmal etwas länger und leider können Sie auch recht wenig dagegen tun. Der Kauf einer – bedauerlicherweise meist recht teuren – Festplatte lindert solche Beschwerden allerdings erheblich.

Hier woilen wir nun die Besprechung der Menüs erst einmal abbrechen und uns den Fenstern des Amiga zuwenden – neben der Maus und den Menüs wohl dem dritten wichtigen Konzept der Amiga-Software.

2.6 Schöne Aussichten: Fenster

Die Fenster des Amiga besitzen einige interessante Eigenschaften, die die Arbeit mit diesem Computer wesentlich erleichtern. Unter anderem ermöglichen sie es, »mehr Informationen gleichzeitig auf dem Bildschirm erscheinen zu lassen als eigentlich Platz haben«. Sollten Sie

Zweifel daran haben, legen Sie dieses Buch nicht als Unsinn beiseite. Die Behauptung steht ja in Anführungsstrichen und ist also mit Vorsicht zu genießen. Ein Quentchen Wahrheit steckt aber trotzdem darin!

2.6.1 Wie Sie Fenster verschieben

Das einfachste, was wir mit einem Fenster machen können, das auf den Open-Befehl hin erschien, ist, es zu bewegen. Die Bewegung eines Fensters geht ganz ähnlich vonstatten wie das Bewegen des Disketten-Piktogramms in einem der vorangegangenen Abschnitte. Während man das Disketten-Symbol aber einfach an beliebiger Stelle irgendwo mit der Maus greifen konnte, sind Fenster etwas wählerischer.

Wichtig für viele Operationen mit einem Fenster ist die »Titelleiste«. Dies ist der breite Streifen am oberen Fensterrand, der unter anderem den Namen des Fensters am linken Rand enthält. Diese Titelzeile samt Fensternamen ist zunächst etwas undeutlich. Klicken Sie auf die blaue Fläche im Innern des Fensters (nicht auf eines der Symbole darin). Das Fenster wird dadurch »zur weiteren Bearbeitung« selektiert. Die vorher schlecht leserliche Schrift des Fenster-Titels wird klar und die kleinen blauen Pünktchen neben dem Namen werden zu zwei blauen Linien. Dies bedeutet, daß das Fenster nun selektiert ist.

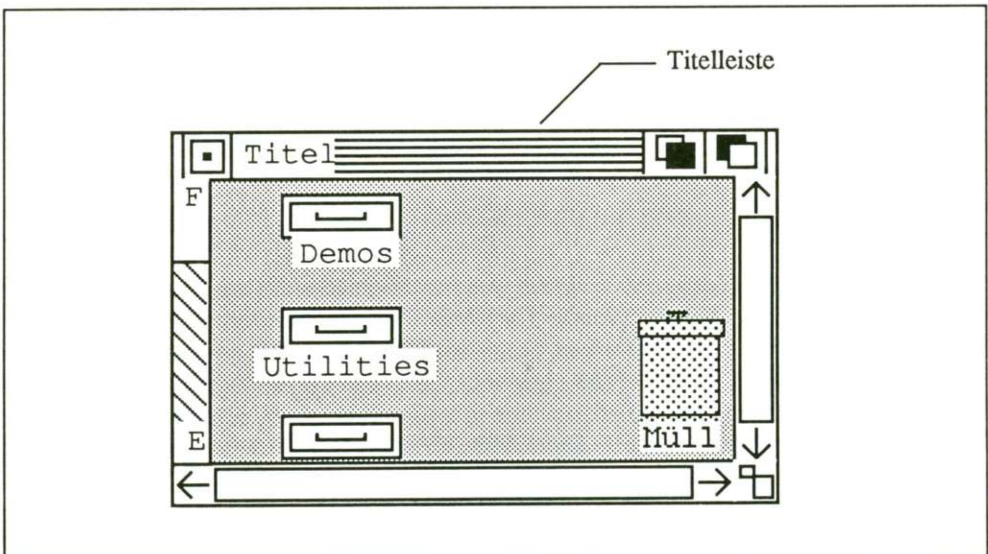


Bild 2.7: Ein Fenster

Wenn Sie jetzt mit der linken Maustaste auf den Namen oder die vier blauen Linien daneben klicken, ergreifen Sie das Fenster – genauso, wie Sie ein Disketten-Piktogramm ergreifen können, indem Sie darauf klicken. Halten Sie die Maustaste fest und bewegen Sie die Maus.

Den Mausbewegungen folgt jetzt ein orangefarbenes Rechteck, welches den Rahmen des Fensters symbolisiert. Lassen Sie die Maustaste in irgendeinem Augenblick los, so springt auch der Rest des Fensters samt Inhalt an die Stelle, an der sich der orangefarbene Rahmen in diesem Augenblick befand.

Experimentieren Sie etwas mit dem Bewegen von Fenstern. Es ist eine Sache, die in fast jedem Programm häufig gebraucht wird. Achten Sie dabei zum Beispiel darauf, daß die Umrißlinie des Fensters immer komplett auf dem Bildschirm bleibt. Sie stößt bei der Bewegung am Bildschirmrand an und auch der Mauszeiger kann dann in dieser Richtung nicht weiter bewegt werden. Fenster müssen immer komplett auf dem Bildschirm sichtbar sein!

2.6.2 Überlappende Fenster

Eine der wichtigsten Eigenschaften der Fenster ist die Fähigkeit, sich zu überlappen. Ein Fenster kann ein anderes ganz oder teilweise verdecken, wie ein Blatt Papier auf dem Schreibtisch ein anderes Blatt ganz oder teilweise verdecken kann. Wendet sich das Interesse wieder einem (teilweise) verdeckten Fenster zu, so zieht man es auf dem Bildschirm einfach zur Seite, damit wieder mehr oder alles davon sichtbar wird.

Um das ausprobieren zu können, brauchen wir natürlich ein zweites Fenster. Dazu klicken wir im Fenster *Workbench* das Schubladen-Piktogramm an, unter dem der Name *System* steht, und wählen dann wieder den Befehl *Open* aus dem Menü Workbench aus. Daraufhin erscheint ein neues kleines Fenster am Bildschirm. Schubladen enthalten – wie es sich gehört – andere Dinge, die sichtbar werden, wenn man sie öffnet. Was das Fenster enthält, welches aus der Schublade *System* hervorgegangen ist, soll uns hier zunächst nicht weiter interessieren. Wir benötigen nur das Fenster selbst. Um noch ein Fenster zu erhalten, öffnen Sie jetzt bitte auch die Schublade *Utilities*.

Nehmen Sie nun abwechselnd die beiden Fenster und schieben Sie sie auf dem Bildschirm hin und her. Sie sehen dabei sehr deutlich, daß jedes dieser Fenster ein »Blatt für sich« ist. Die Symbole des Fensterinhalts, die verdeckt wurden, erscheinen sofort wieder, sobald der entsprechende Teil des Fensters wieder freigelegt wird. Spinnen wir den Vergleich von Fenstern mit Blättern Papier auf einem Schreibtisch etwas weiter, so müßte es auch bei Fenstern eine Möglichkeit geben, eines ganz oben auf »den Stapel« oder »nach hinten« zu legen. Bis jetzt können Sie Fenster ja nur gegeneinander verschieben. Aber auch die Reihenfolge der Fenster »im Stapel« kann auf dem Amiga geändert werden.

2.6.3 Fenster-Gadgets

Zum Ändern der Reihenfolge der Fenster »im Stapel« dienen die beiden kleinen Symbole am rechten Rand der Titelleiste. Es sind sogenannte *Gadgets* (zu deutsch etwa *Geräte* oder *Instrumente*), die, ähnlich wie eine Menüauswahl, einen Befehl an ein Programm schicken, wenn sie angeklickt werden. Das linke der beiden Symbole heißt *Back-Gadget* und legt ein Fenster »ganz nach hinten«, das rechte Symbol heißt *Front-Gadget* und kann ein Fenster »ganz nach oben« holen.

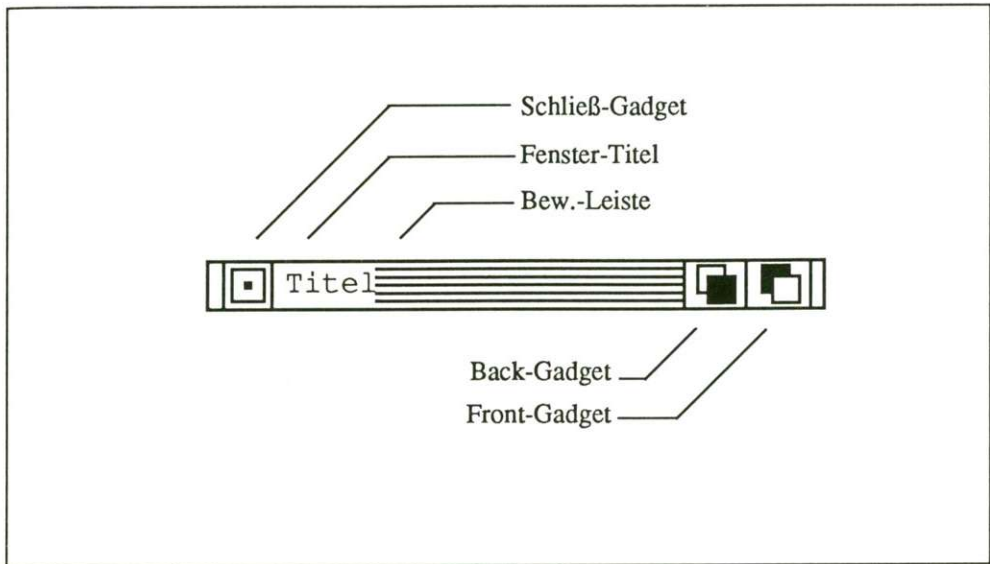


Bild 2.8: *Die Titelleiste eines Fensters*

Verteilen Sie die drei Fenster nun so auf dem Bildschirm, daß sie sich gegenseitig übderdecken, aber von jedem wenigstens ein »Zipfelchen« der Titelleiste zu sehen ist. Nun können Sie die Gadgets in Ruhe ausprobieren. Klicken Sie in das Back-Gadget und das Fenster fällt nach hinten. Alle Teile anderer Fenster, die von diesem verdeckt wurden, werden wieder sichtbar. Ist das Front-Gadget eines teilweise verdeckten Fensters sichtbar, so klicken Sie darauf und beobachten Sie, wie es ganz nach oben kommt und sein kompletter Inhalt sichbar wird.

Interessiert Sie der Inahlt eines Fensters überhaupt nicht mehr, so können Sie das Fenster ganz schließen. Hierzu können Sie natürlich den Befehl *Close* verwenden, der jedes Fenster wieder schließen kann, das mit *Open* geöffnet wurde. Es gibt allerdings auch eine »Abkürzung«, die Ihnen den ach so langen Weg in die Menüzeile spart. Klicken Sie in das kleine quadratische Gadget am linken Rand der Titelzeile eines Fensters, so wirkt dies, als hätten Sie das Fenster selektiert und dann den Befehl *Close* ausgewählt. Dieses Gadget heißt *Schließ-Gadget* (oder englisch *Close-Gadget*).

Allen Gadgets, die Sie bis jetzt kennengelernt haben, ist eine Eigenschaft gemeinsam, die Sie sofort austesten können. Diese Gadgets bewirken nämlich nur dann etwas, wenn Sie die linke Maustaste niederdrücken, während sich der Mauszeiger über dem Gadget befindet, und die Taste dann aber auch wieder loslassen. Probieren Sie es einfach aus. Wenn Sie zum Beispiel in ein Front-Gadget klicken und die Aktionstaste festhalten, können sie sehen, wie sich die Färbung des Gadgets ändert. Bewegen Sie den Mauszeiger bei gedrückter Taste vom Gadget fort, erhält es die ursprüngliche Färbung wieder. Wenn Sie jetzt die Taste loslassen, geschieht gar nichts. Bewegen Sie aber vor dem Löschen die Taste der Maus wieder über das Gadget,

so schlägt dessen Farbe wieder um, und wenn Sie jetzt die Taste loslassen, »wirkt das Gadget« und das Fenster kommt nach oben.

Somit haben Sie jetzt mehrere übereinanderliegende Blätter auf dem Bildschirm, von denen jedes fast so groß werden kann wie ein ganzer Bildschirm und von denen Sie beliebig eines nach oben holen oder auch ganz beiseite legen (schließen) können. Die Behauptung, mit Fenstern könne man mehr Informationen auf dem Bildschirm zeigen, als eigentlich Platz haben, klingt damit also weniger absurd.

Fenster können noch eine ganze Menge anderer Eigenschaften haben. Für dieses Kapitel haben wir uns aber lange genug mit Fenstern beschäftigt. Zum Abschluß des Kapitels sollen Sie jetzt noch eine ganz elementare Operation kennenlernen, die ein Computerbesitzer gar nicht oft genug durchführen kann.

2.7 Sicher ist sicher: Kopien

Eine Sache, die jeder Anwender eines Computers möglichst zu Anfang seines Kontakts mit einem neuen Modell erlernen sollte, ist das Anfertigen von Sicherungskopien. Die 3,5-Zoll-Disketten des Amiga sind zwar ein relativ robustes Speichermedium – aber bekanntlich hält nichts ewig. Bereits eine kleine Unachtsamkeit Ihrerseits oder ein kleiner Materialfehler können ihr schaden. Unter den Begriff Unachtsamkeit würde zum Beispiel fallen, wenn Sie eine Diskette auf einen Lautsprecher, ein Gerät mit einem starken Elektromotor, unter ein Telefon oder auf die Heizung legen. Am gefährlichsten sind alle Ablagen, bei denen die Diskette einem starken magnetischen Feld ausgesetzt wird (wie es Elektromotoren und Telefonklingeln zum Beispiel erzeugen). Aber auch die Alterung kann eine Rolle spielen.

Die möglichen Schäden, die dabei auftreten, können unter Umständen in kleinem Rahmen bleiben. Sie könnten zum Beispiel nur ein Dokument betreffen, das Sie auf einer Diskette gespeichert haben, es könnte aber auch eine ganze Diskette unbrauchbar werden, so daß alle Daten darauf verlorengehen. Wenn Sie dann keine Sicherungskopie dieser Diskette haben, ist möglicherweise die Arbeit von Wochen verloren.

Deshalb gilt die Grundregel Nummer 1: Man kann gar nicht oft genug Kopien seiner wichtigen Disketten anlegen!

Dies gilt in dieser Form natürlich nur für Disketten, bei denen sich die darauf gespeicherten Daten andauernd ändern. Eine Diskette wie die Workbench-Diskette, die Sie nie oder nur sehr selten ändern, braucht nur einmal beziehungsweise einmal nach jeder größeren Änderung kopiert zu werden. Üblicherweise verfährt man dabei so, daß man nach dem Kopieren das Original an einer sicheren Stelle unterbringt (am besten so, daß auch neugierige Kinder nicht daran kommen können) und mit der Kopie weiterarbeitet. Man spricht in diesem Zusammenhang oft auch von einer *Arbeitskopie*.

Von besonders wichtigen oder schwer wieder zu beschaffenden Disketten sollten Sie sogar mindestens 2 Kopien anlegen.

2.7.1 Vorbereitungen

Bevor Sie mit dem eigentlichen Kopieren anfangen, müssen Sie zunächst einige Vorbereitungen treffen. Legen Sie die *Original-Diskette,* also die Diskette, die Sie kopieren wollen, bereit und sehen Sie nach, ob sie eindeutig beschriftet ist. »Eindeutig« heißt in diesem Fall, daß die Beschriftung den Inhalt deutlich beschreibt und sie nicht mit einer anderen Diskette verwechselt werden kann. Falls dies nicht der Fall ist, holen Sie die eindeutige Beschriftung nach – zum Beispiel mit einem permanenten (!) Faserschreiber.

Aktivieren Sie nun den Schreibschutz der Original-Diskette! Schieben Sie dazu den kleinen Plastik-Schieber zum Rand der Diskette, so daß Sie durch das entstehende Loch hindurch, schauen können. (Der Schreibschutz-Schieber befindet sich in der rechten oberen Ecke der Diskette, wenn Sie auf die Diskettenoberseite schauen und der Metallschieber nach unten zeigt.) Dieser Schreibschutz macht es physikalisch unmöglich, daß bei einem Fehler während des Kopiervorgangs irgend etwas am Original zerstört wird.

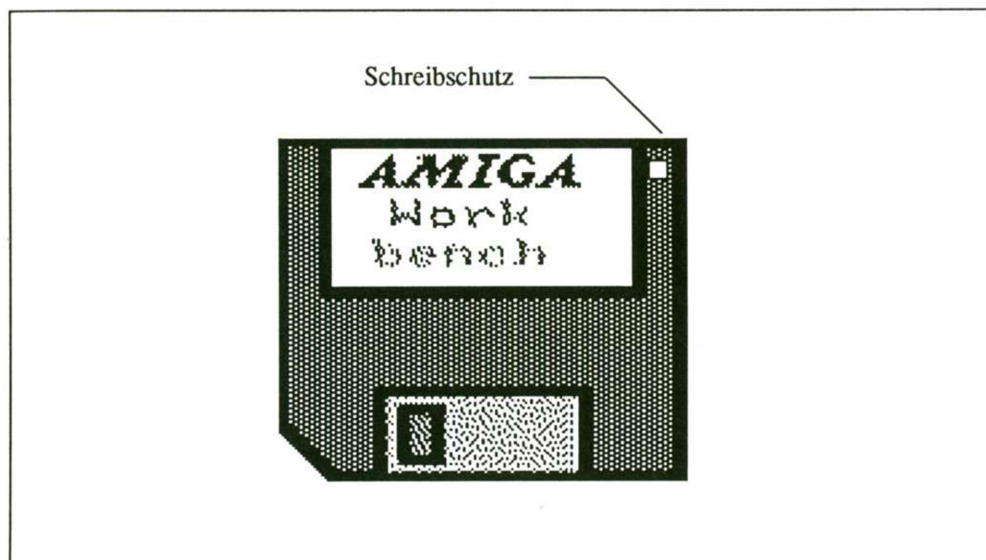


Bild 2.9: *Eine Diskette und der Schreibschutz-Schieber*

Legen Sie dann die Diskette bereit, die die Kopie aufnehmen soll. Es kann eine neue Diskette frisch aus der Packung sein oder eine schon gebrauchte, deren Inhalt nicht mehr wichtig ist. Versehen Sie sie mit exakt derselben Beschriftung wie die Original-Diskette und setzen Sie noch groß KOPIE dahinter (oder BACKUP, wenn Sie professionell wirken wollen). Achten Sie darauf, daß der Schreibschutz der Kopie deaktiviert ist! Der Schreibschutz-Schieber muß also in einer Position stehen, die das kleine Loch in der entsprechenden Diskettenecke verdeckt, so daß Sie nicht mehr hindurchschauen können.

Nun sind Sie bestens gerüstet, für »die Dinge, die da kommen mögen«. Sind Sie den Anweisungen exakt gefolgt, so können Sie keinen Fehler machen, der Ihre wertvolle Original-Diskette beschädigt.

Die Anfertigung einer Sicherungskopie läuft auf der Workbench ab. Starten Sie deshalb nun die Workbench, wenn Sie sich noch nicht auf ihr befinden. Falls Sie sich bereits auf der Workbench befinden, schließen Sie alle Fenster (und beenden Sie alle Programme, sofern Sie welche gestartet haben).

Der genaue Ablauf des Kopiervorgangs hängt nun stark davon ab, ob Sie ein oder zwei Diskettenlaufwerke besitzen. Naturgemäß ist es mit einem zweiten Laufwerk einfacher. Vielleicht ist das ein Anreiz für Sie, sich ein solches anzuschaffen.

2.7.2 Kopieren mit einem Diskettenlaufwerk

Besitzen Sie nur ein Diskettenlaufwerk, so steht Ihnen eine kleine Tortur bevor. Das Kopieren mit einem Diskettenlaufwerk ist kein reines Vergnügen.

Falls Sie nicht die Workbench-Diskette selbst kopieren wollen, werfen Sie diese zunächst erst einmal aus. Halten Sie sie aber bereit; sie wird gleich wieder gebraucht. Schieben Sie die Diskette ein, die Sie kopieren wollen und warten Sie, bis das entsprechende Symbol am Bildschirm erscheint.

Selektieren Sie nun die eben eingelegte *Original-Diskette* (mit der linken Maustaste) und drücken Sie dann die Menütaste der Maus. Gehen Sie mit dem Mauszeiger über das Wort *Workbench* in der Menüleiste und fahren Sie dann – bei immer noch gedrückter Taste – über den Befehl *Duplicate*. (Falls dieser grau ist und nicht aktiviert werden kann, ist die Diskette nicht mehr selektiert. Klicken Sie erneut auf das Piktogramm der zu kopierenden Diskette.) Lassen Sie die Maustaste los, wenn das Wort *Duplicate* orange auf schwarz erscheint.

Falls Sie nicht die Workbench-Diskette kopieren, erscheint jetzt ein Fenster am oberen linken Bildschirmrand mit der Aufforderung, die Workbench-Diskette wieder einzulegen. Werfen Sie in diesem Fall die zu kopierende Diskette aus und schieben Sie statt dessen die Workbench-Diskette ein. Das Fenster verschwindet daraufhin und das Diskettenlaufwerk arbeitet einen Moment lang. Dann werden Sie wieder aufgefordert, die Diskette, die sie kopieren wollen, einzulegen.

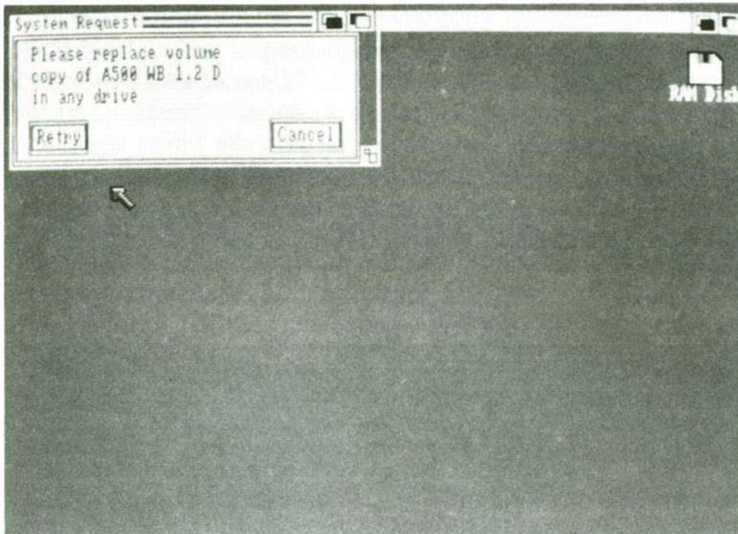


Bild 2.10: Aufforderung zum Einlegen der Workbench-Diskette

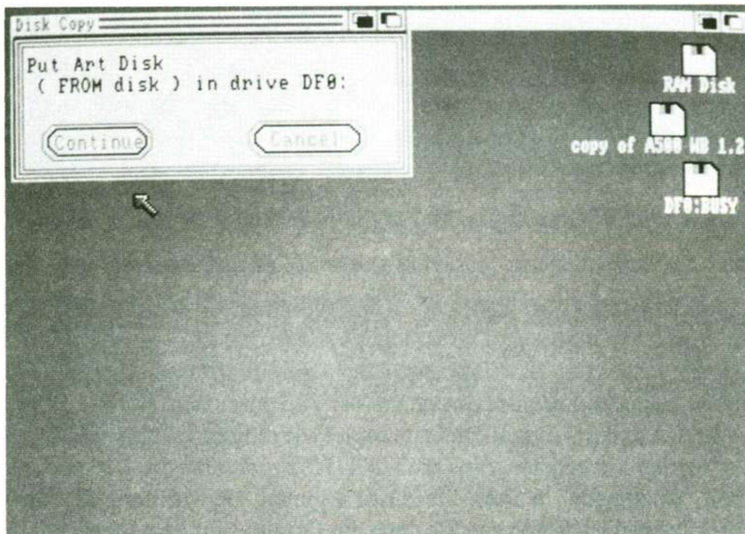


Bild 2.11: Aufforderung zum Einlegen der FROM-Diskette

Es erscheint nun ein neues Fenster namens *Disk Copy* mit der Aufforderung, die *FROM disk* ins Diskettenlaufwerk einzulegen. Mit *FROM disk* ist immer die Originaldiskette, also die zu kopierende Diskette, gemeint. Am unteren Rand des Fensters *Disk Copy* befinden sich zwei abgerundete Rechtecke – sogenannte Gadgets – in Form von schwarz und orange umrahmten Worten. Diese Gadgets verhalten sich genauso wie Tasten an einem Radio oder Kassettenrecorder. Das eine lautet *Continue* (engl. für *Fortfahren*), das andere *Cancel* (engl. für *Abbrechen*). Wenn Sie sicher sind, daß die richtige Diskette im Laufwerk liegt, klicken Sie auf *Continue*. Falls nicht, so klicken Sie bitte auf *Cancel*, wodurch der gesamte Kopiervorgang abgebrochen wird.

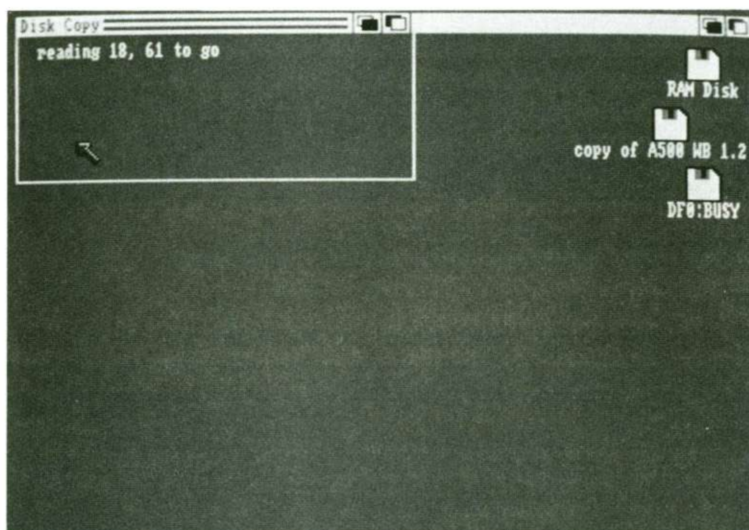


Bild 2.12: *Protokollierung des Kopiervorgangs*

Haben Sie *Continue* geklickt, so startet der Kopiervorgang. Im Fenster *Disk Copy* wird der Fortschritt des Vorgangs angezeigt. Die Meldung *reading 23, 78 to go* bedeutet zum Beispiel, daß bereits 23 von insgesamt 78 »Häppchen« der Diskette gelesen wurden. (Für den Fachmann: Bei diesen Häppchen handelt es sich um die 78 Spuren der Diskette.) Der Amiga liest nun nämlich vom Inhalt der Original-Diskette soviel wie möglich in seinen RAM-Speicher. Ist dieser voll, so werden Sie gebeten, die *DESTINATION disk*, also die Diskette, die die Kopie aufnehmen soll, einzulegen. Warten Sie einen Moment und achten Sie darauf, daß das Diskettenlicht erloschen ist. Entfernen Sie dann das Original aus dem Laufwerk und schieben die Diskette ein, die die Kopie aufnehmen soll. Wenn Sie damit fertig sind, klicken Sie wieder auf *Continue*!

Der Amiga schreibt nun die Daten von der Original-Diskette, die er in seinem RAM gespeichert hat, auf die Kopie. Auch dieser Vorgang wird wieder protokolliert. Die Meldung *writing 23, 78 to go* bedeutet zum Beispiel, daß bereits 23 von insgesamt 78 »Häppchen« auf die zweite Diskette geschrieben wurden. Nach kurzer Zeit fordert Sie der Amiga wieder auf, die Original-Diskette (FROM disk) einzulegen, damit er die nächsten Häppchen davon lesen kann. Folgen Sie dieser Aufforderung und klicken dann wieder auf *Continue*!

Dieses Wechseln zwischen Originaldiskette (FROM disk) und Kopie (DESTINATION disk) wird noch einige Male nötig werden. Beim Amiga 500 ohne Speichererweiterung sind etwa 10 bis 12 Diskettenwechsel nötig; beim Amiga mit Speichererweiterung sind es nur etwa 5. Achten Sie dabei immer auf den Text im Fenster. Wird die *FROM disk* verlangt, müssen Sie das Original einlegen. Wird die *DESTINATION disk* verlangt, müssen Sie die Kopie einlegen. Achten Sie beim Herausnehmen der Disketten immer darauf, daß das rote Diskettenlämpchen aus ist und das Laufwerk keine hörbaren Geräusche von sich gibt. Wenn Sie mit dem Diskettenwechsel fertig sind, müssen Sie jedesmal auf *Continue* klicken (oder auf *Cancel*, wenn Sie die Tortur abbrechen wollen).

2.7.3 Kopieren mit zwei Diskettenlaufwerken

Diskettenkopieren mit zwei Laufwerken ist wesentlich angenehmer. Legen Sie dazu die Original-Diskette ins interne Laufwerk und die Diskette, die die Kopie aufnehmen soll, ins externe Laufwerk. Falls die zweite Diskette neu und bislang unbenutzt ist, erscheint sie unter dem Namen *DF1:BAD*. Klicken Sie dann mit der linken Maustaste auf das Piktogramm der Original-Diskette und ziehen den Zeiger dann über das Piktogramm der Kopie. Nun lassen Sie die Maustaste los. Dieses Legen der einen Diskette auf die andere sagt der Workbench, daß Sie eine Kopie mit zwei Laufwerken machen wollen. Falls auf Ihrer Workbench noch ein anderes Piktogramm namens *RAM Disk* erscheint, achten Sie bitte darauf, das Piktogramm der zu kopierenden Diskette nicht auf *RAM Disk* zu legen. Mit dieser Diskette hat es eine besondere Bewandtnis. Nähere Erläuterungen zum Thema RAM-Disk finden Sie im zweiten Teil dieses Buches (ab Kapitel 7), der sich hauptsächlich mit dem CLI beschäftigt.

Falls Sie nicht die Workbench-Diskette selbst kopieren, erscheint jetzt ein Fenster mit der Aufforderung, die Workbench einzulegen. Werfen Sie dazu die Original-Diskette aus und legen Sie die Workbench-Diskette ein. Nach wenigen Augenblicken verschwindet diese Aufforderung und der Amiga verlangt wieder nach der Originaldiskette. Tauschen Sie daraufhin wieder die Workbenchdiskette gegen die Originaldiskette aus. Dann erscheint ein neues Fenster namens *Disk Copy*.

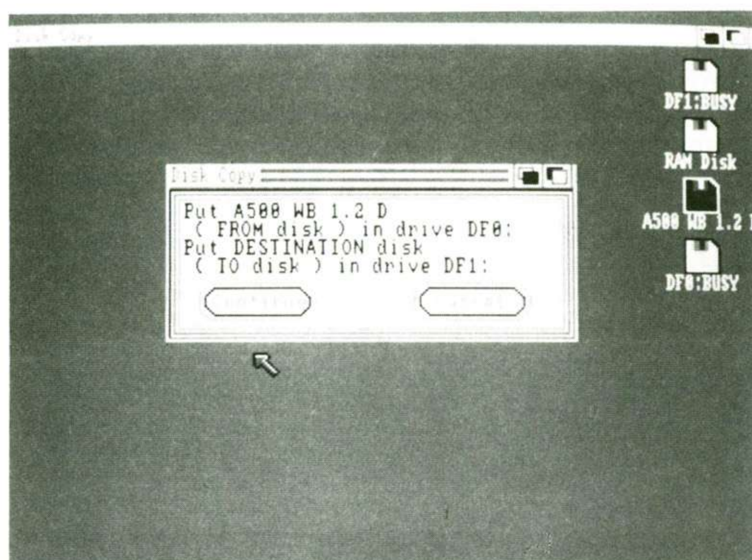


Bild 2.13: *Das Disk-Copy-Fenster beim Kopieren mit zwei Laufwerken*

In diesem Fenster erscheint unter anderem die Aufforderung, das Original (FROM disk) ins interne Laufwerk (drive DF0:) und die Kopie (DESTINATION disk) ins externe Laufwerk (drive DF1:) einzulegen. Tun Sie das jetzt und vergewissern Sie sich auf alle Fälle noch einmal davon, daß in jedem Laufwerk die richtige Diskette liegt! Am unteren Rand des Fensters *Disk Copy* befinden sich zwei schwarz und orange umrahmte Worte, die Druckknöpfe – sogenannte Gadgets – darstellen, und die sich genauso verhalten wie eine Taste an einem Radio oder Kassettenrecorder. Das eine Wort lautet *Continue* (engl. für *Fortfahren*), das andere *Cancel* (engl. für *Abbrechen*). Wenn Sie sicher sind, daß in jedem Laufwerk die richtige Diskette liegt, klicken Sie auf *Continue*. Falls nicht, so klicken Sie bitte auf *Cancel*, was den gesamten Kopiervorgang abbricht.

Nachdem Sie auf *Continue* geklickt haben, beginnt der Kopiervorgang. Er dauert insgesamt etwa 100 Sekunden. Sie können den Fortschritt im Fenster *Disk Copy* beobachten, um ungefähr zu wissen, wie lange es noch dauert. Es erscheinen abwechselnd Meldungen der Form *reading 23, 78 to go* und *writing 23, 78 to go*. Dies bedeutet, daß der Amiga bereits 23 von 78 nötigen »Häppchen« (Spuren) der Originaldiskette gelesen beziehungsweise schon auf die Kopie geschrieben hat.

2.7.4 Abschluß des Kopiervorgangs

Am Ende des Ganzen steht dann eine komplette Kopie Ihres kostbaren Originals. Der alte Inhalt der Diskette, die die Kopie enthält, wurde dabei allerdings vollständig zerstört! Die Kopie erscheint übrigens auch unter dem Namen *copy of <Originalname>* auf der Workbench.

Auf Wunsch können Sie diesen Namen auch ändern. Den dazu notwendigen Befehl lernen Sie im folgenden Kapitel kennen.

Falls es sich beim Original um die Kickstart-, Workbench- oder die Original-Diskette eines Programms handelt, legen Sie die Diskette beiseite und arbeiten Sie ab jetzt mit der Kopie weiter. Handelt es sich um eine Daten-Diskette, deren Inhalt häufig geändert wird, legen Sie die Kopie beiseite und arbeiten Sie mit dem Original weiter.

Es gibt allerdings einige kommerzielle Programme, die Sie auf diese Weise nicht kopieren können. Diese Disketten haben einen Kopierschutz, der sie vor unautorisiertem (und leider auch vor rechtmäßigem) Kopieren schützt. Es ist traurig, daß Software-Hersteller meinen, zu solchen Methoden greifen zu müssen, aber man kann vorläufig wenig dagegen tun – außer solche Programme zu boykottieren! Es gibt jedoch auch einige Spezial-Kopierprogramme auf dem Markt, denen solche Kopierschutzverfahren nichts ausmachen. (Zwei Beispiele für solche Programme sind die Produkte »Maurauder« und »Mirror«, die Sie bei jedem guten Amiga-Händler erwerben können.) Ein gutes Kopierprogramm dieser Art gehört zur Grundausstattung jeder Softwarebibliothek, denn auch nur vorübergehender erzwungener Verzicht auf ein wichtiges Programm kann unter Umständen große Umstände (und Kosten) bereiten.

3 Objekte der Workbench

Nachdem Sie im Kapitel 2 einen Überblick über die Funktionsweise der Amiga-Werkbank (beziehungsweise *Workbench*) bekommen haben, wollen wir nun »zur Sache« kommen und uns die Objekte auf der Werkbank einmal näher betrachten. Dies sind die verschiedenen Symbole, die auf dem blauen Bildschirmhintergrund und in den Fenstern der Workbench zu sehen sind. Wir nennen sie Piktogramme, und jedes von ihnen repräsentiert ein Objekt, das von der Workbench aus auf irgendeine Weise manipuliert werden kann. Diese Piktogramme entsprechen den Disketten, Programmen und Dateien, mit denen Sie beim Umgang mit dem Amiga täglich zu tun haben werden. Besonders wichtig sind auch noch die Schubladen, die Ihnen helfen, Ordnung auf Ihren Disketten zu halten.

Die im folgenden beschriebenen Befehle und Operationen sollten Sie bei einem ersten Lesen dieses Buches am besten immer sofort ausprobieren. Verwenden Sie dazu sicherheitshalber eine Kopie und nicht die Original-Workbench-Diskette. Wie Sie eine solche Kopie anlegen können, haben Sie ja im letzten Kapitel erfahren.

3.1 Allgemeine Eigenschaften von Piktogrammen

Das erste Piktogramm, welches Sie auf der Workbench kennengelernt haben, war das einer Diskette. Die folgende Abbildung zeigt es Ihnen noch einmal in etwas vergrößerter Form.

***Bild 3.1:** Ein Disketten-Piktogramm*

3.1.1 Wie Sie ein Piktogramm auswählen

»Normalerweise« erscheint ein Disketten-Piktogramm weiß auf blauem Grund mit einem orangen Fleck am oberen Rand, der den Metallschieber einer 3,5-Zoll-Diskette andeuten soll. Klicken Sie es aber mit der linken Maustaste an, so wird es schwarz und der orangefarbene Klecks wird blau. Jedes Piktogramm reagiert so oder so ähnlich darauf, wenn es mit der Maus ausgewählt beziehungsweise selektiert wurde.

3.1.2 Wie Sie ein Piktogramm verschieben

Halten Sie die linke Maustaste fest, nachdem Sie das Piktogramm angeklickt haben, so können Sie es bewegen. Dazu bewegen Sie einfach den Mauszeiger zu der gewünschten Stelle und lassen die Taste dann wieder los. Aber dies kennen Sie ja schon aus Kapitel 2. Diesen Vorgang des »Ergreifens« und Verschiebens eines Piktogramms mit der Maus nennen wir in Zukunft *Verschieben* oder *Legen eines Piktogramms* (an eine bestimmte Stelle). Die Mausbewegungen und Maustastenbetätigungen, die dazu nötig sind, werde ich nicht jedesmal ausführlich beschreiben.

Ein Piktogramm bleibt so lange an der Stelle liegen, an der Sie es zuletzt hingelegt haben, bis Sie die Workbench verlassen. Das Verlassen der Workbench ist normalerweise nur durch ein Abschalten oder Neustarten (siehe Kapitel 2) des Amiga zu erreichen. Nach dem Abschalten »vergißt« das Piktogramm seine neue Position. Beim nächsten Start der Workbench liegt es wieder an der alten Position. Möchten Sie, daß ein bestimmtes Piktogramm stets an einer anderen Position erscheint, so können Sie auch dies erreichen. Was Sie dazu tun müssen, wird weiter unten noch genauer beschrieben.

3.1.3 Wie Sie ein Piktogramm umbenennen

Unter jedem Piktogramm, das Sie am Bildschirm sehen, steht ein Name. Dieser dient vor allem dazu, Piktogramme mit demselben Aussehen unterscheiden zu können. Den Namen, beziehungsweise den Text, der unter dem Piktogramm erscheint, können Sie jederzeit ändern. Selektieren Sie dazu zum Beispiel ein Disketten-Piktogramm und wählen dann den Befehl *Rename* (engl. *to rename = umbenennen*) aus dem Menü *Workbench*. Mitten auf dem Bildschirm erscheint daraufhin ein schmaler, weiß umrandeter Streifen, in dem der Name der Diskette steht (zum Beispiel »Workbench«). Klicken Sie mit der linken Maustaste in diesen Streifen und Sie können den Text ändern.

Der erste Buchstabe des Namens erscheint zunächst einmal schwarz auf orange, während alle anderen Buchstaben weiß auf blau dargestellt werden. Der schwarze Buchstabe markiert die aktuelle Schreibmarke (der Fachmann sagt dazu auch Cursor). Wenn Sie auf der Tastatur nun zu tippen beginnen, erscheint der neue Text immer **vor** dieser Schreibmarke und die dahinterstehenden Buchstaben werden automatisch nach rechts geschoben. Sie können den Buchstaben, auf dem die Schreibmarke steht, auch löschen. Drücken Sie hierzu die [Del]-Taste oben rechts auf der Tastatur. Der Buchstabe bei der Schreibmarke verschwindet und die Buchstaben rechts davon rutschen dadurch nach links. Wollen Sie einen völlig neuen Namen eingeben, so

tippen Sie einfach ein paarmal auf [Del], bis der alte Name völlig verschwunden ist und geben dann den neuen ein.

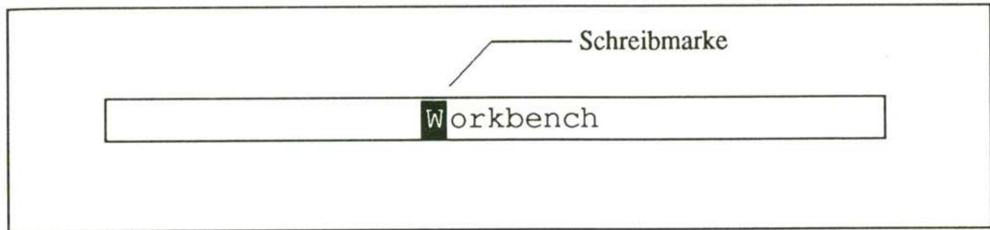


Bild 3.2: *Ein Rahmen für die Texteingabe*

Wenn Sie einen Tippfehler machen, können Sie immer den letzten eingegebenen Buchstaben mit der [Backspace]-Taste (links neben der [Del]-Taste) löschen. Die [Backspace]-Taste löscht aber nicht nur den zuletzt eingetippten Buchstaben, sondern immer den Buchstaben links von der Schreibmarke, so daß Sie also auch mehrere Buchstaben auf diese Weise wieder löschen können.

Rechts unten auf der Tastatur befinden sich weiterhin vier Tasten, die je einen Pfeil tragen. Dies sind die Cursortasten, die Sie zum Verschieben der Schreibmarke verwenden können. Die beiden nach links und rechts weisenden Pfeile bewegen die Schreibmarke jeweils einen Buchstaben nach links beziehungsweise rechts, sofern dies möglich ist. (Die beiden Pfeile nach oben und unten haben beim Ändern eines Piktogrammnamens keine Wirkung.) Statt mit diesen beiden Tasten können Sie die Schreibmarke aber auch mit der Maus bewegen. Klicken Sie dazu einfach den Buchstaben mit der Spitze des Mauszeigers an, zu dem Sie die Schreibmarke bewegen wollen.

Sind Sie mit dem Namen in dem weißen Rahmen zufrieden, so tippen Sie kurz auf die [Return]-Taste, und das zuvor ausgewählte Disketten-Piktogramm erhält den neuen – gerade eingegebenen – Namen. (Falls die Diskette schreibgeschützt war, erhält das Icon keinen neuen Namen, und ein Fenster mit einer Fehlermeldung erscheint. Dazu mehr im folgenden Abschnitt *Wie Fehlermeldungen aussehen*. Die [Return]-Taste schließt typischerweise immer die Eingabe von Text in einen solchen Rahmen ab! Sie sollten sich das für später merken. Auch andere Befehle oder Programme verlangen manchmal Texteingaben, die immer ähnlich ablaufen.

3.1.4 Wie Fehlermeldungen aussehen

Der Befehl *Rename* bietet eine gute Gelegenheit, auch einmal die Fehlermeldungen des Amiga kennenzulernen. Fehlermeldungen zu erkennen und ihre Ursachen zu beheben, ist ein wichtiger Bestandteil der Arbeit mit dem Computer – denn Fehler macht schließlich jeder einmal.

Nehmen Sie dazu nun die Workbench-Diskette aus dem Laufwerk und aktivieren Sie den Schreibschutz (indem Sie den kleinen Plastikschieber zum Rand der Diskette schieben). Legen Sie die Diskette dann zurück in das Laufwerk, wählen Sie das Piktogramm mit der Maus aus und wählen Sie den Befehl *Rename* aus dem Menü *Workbench*. Geben Sie ein paar Buchstaben ein und drücken Sie auf die [Return]-Taste. Sofort erscheint ein neues Fenster auf dem Bildschirm, das Ihnen mitteilt, daß die Diskette schreibgeschützt (engl. *write protected*) ist.

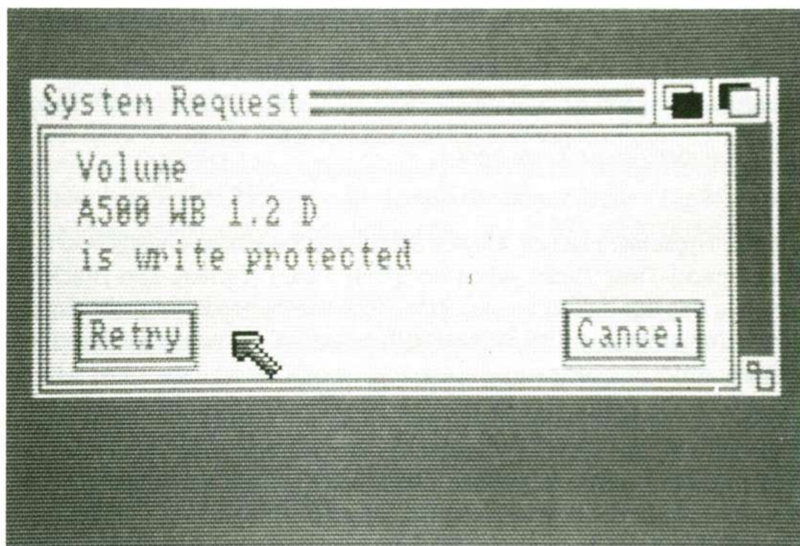


***Bild 3.3:** Die erste Fehlermeldung*

Dieses Fenster ist ein sogenannter *Requester*. Es verlangt eine Reaktion von Ihnen (engl. *to request = verlangen*). Hierzu enthält es zwei Knöpfe, die Sie mit dem Mauszeiger betätigen können. Wenn Sie *Cancel* anklicken, wird der *Rename*-Befehl abgebrochen und die Diskette behält den alten Namen. Sie können aber auch die Disketten aus dem Laufwerk nehmen, den Schreibschutz entfernen und dann auf *Retry* (engl. für *versuche es noch einmal*) klicken. Dann erhält die Diskette den neuen Namen.

Wenn eine Fehlermeldung auf dem Bildschirm erscheint und Sie *Cancel* drücken, erscheint in der Titelzeile noch einmal eine kurze Meldung, die verschwindet, sobald Sie eine Maustaste drücken. Diese Meldung beginnt mit einer Zahl, die den aufgetretenen Fehler eindeutig identifiziert. Falls Sie einmal nicht wissen, was Sie falsch gemacht haben oder wie Sie den Fehler beheben, können Sie mit Hilfe der Fehlernummer im Anhang dieses Buches nachschlagen. Dort sind alle möglichen Fehlermeldungen mit Erläuterungen und Lösungsvorschlägen aufgelistet.

3.1.5 Weitergehende Informationen über ein Piktogramm

Wenn Sie ein Piktogramm auf der Workbench sehen, kennen Sie zunächst einmal nur seinen Namen und können aus seinem Aussehen vielleicht darauf schließen, um welche Art von Piktogramm es sich handelt. Diese Informationen werden Ihnen auch meist genügen. Wenn Sie aber detaillierte Informationen über ein Piktogramm wünschen, so können Sie diese mit dem Befehl *Info* erhalten. Selektieren Sie dazu das Piktogramm, über das Sie mehr Informationen erhalten wollen (zum Beispiel eine Diskette), und wählen Sie dann den Befehl *Info* aus dem Menü *Workbench*. Daraufhin erscheint ein neues Fenster am Bildschirm, das den Namen *Info Release 1.2* trägt.

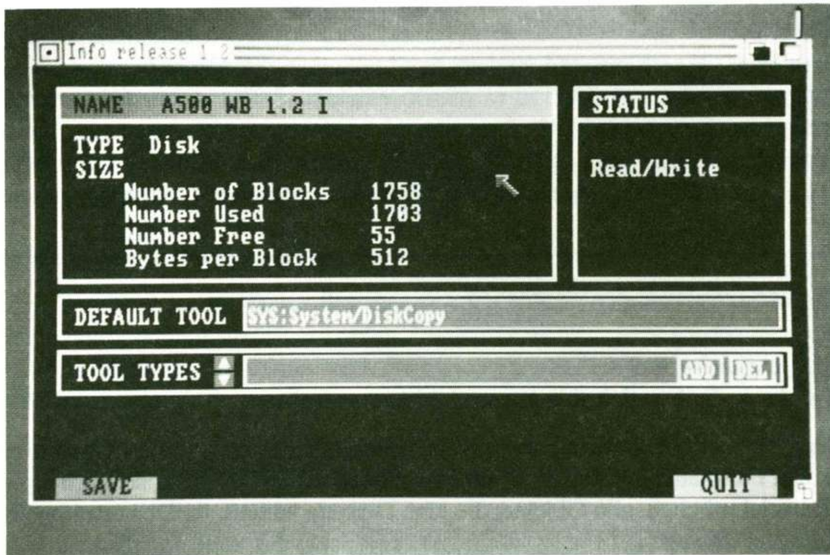


Bild 3.4: Das Info-Fenster einer Diskette

In diesem Fenster wird unter anderem noch einmal der Name des Piktogramms aufgelistet und darunter, neben dem Wort TYPE, erfahren Sie, um welche Art Piktogramm es sich handelt, welchen Typ (engl. *type*) es besitzt. Abhängig von diesem Typ tauchen darunter noch eine ganze Reihe weiterer Informationen auf. Bei einem Disketten-Piktogramm sind dies zum Beispiel ein paar Zahlen, die Aufschluß darüber geben, wieviele Daten sich schon auf der Diskette befinden und wieviel Platz noch frei ist (mehr darüber weiter unten).

Das Info-Fenster können Sie wieder verschwinden lassen, indem Sie sein Schließ-Gadget in der oberen linken Ecke oder den Knopf *Quit* in der rechten unteren Ecke des Fensters mit der Maus anklicken. Den Knopf *Save* sollten Sie nicht betätigen. Er ist nur sinnvoll, wenn Sie die Felder neben *Default Tool* oder *Tool Type* modifiziert haben (was Sie vorläufig auf keinen Fall tun sollten).

3.2 Ordnung auf der Werkbank: Schubladen

Nach diesen mehr allgemeinen Eigenschaften von Piktogrammen werden wir uns nun mit einer besonderen Sorte von Piktogrammen beschäftigen, den Schubladen. Man kann sagen, daß die Schubladen die wichtigsten Objekte sind, die sich auf der Workbench befinden. Wir haben die Schubladen ja schon kurz in Kapitel 2 kennengelernt, als wir Fenster zum Experimentieren brauchten. In Schubladen (und Disketten-Piktogrammen) verbergen sich immer andere Objekte. Die Schubladen sind an sich keine »richtigen« Objekte; sie enthalten nur andere Objekte. (Schubladen entsprechen auf einer nicht ganz so bildlichen Ebene den *Directories* oder *Dateiverzeichnissen*, die Sie in den Kapiteln über das CLI noch kennenlernen werden. Wenn Sie schon mit MS-DOS- oder UNIX-Computern gearbeitet haben, wird Ihnen dieses Konzept schon vertraut sein.)

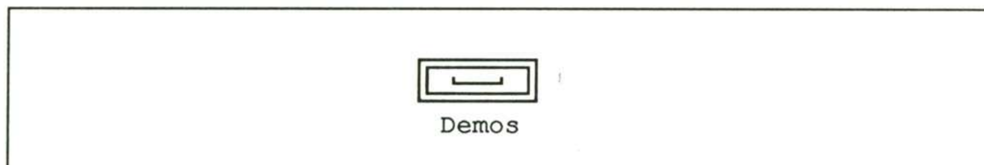


***Bild 3.5:** Eine Schublade der Workbench*

3.2.1 Welche Aufgaben Schubladen und Disketten haben

Schubladen dienen der Gliederung der Informationen beziehungsweise der Objekte auf einer Diskette oder Festplatte. Sie sind nicht unbedingt notwendig, aber praktisch. Prinzipiell wäre es zum Beispiel möglich, alle Objekte, die eine Diskette enthält, im Diskettenfenster selbst unterzubringen. Solange es 10 oder 20 Objekte sind, mag das völlig ausreichen. Wenn aber einige hundert oder (bei Verwendung einer Festplatte) gar einige tausend Objekte auf einer Diskette (beziehungsweise Festplatte) liegen, wird man rasch die Übersicht verlieren und nur sehr schwer ein bestimmtes Objekt finden, das man gerade sucht.

Auf einer richtigen Werkbank würden Sie ja auch nicht alle Gegenstände einfach herumliegen lassen, sondern sie in Kisten, Kästchen und Schubladen legen, die jeweils eine zusammengehörige Gruppe von Gegenständen enthält. Eine Kiste für die Bohrer, eine für Nägel, Schrauben, Dübel, Schraubenschlüssel, Schraubendreher und so weiter. Genauso funktionieren die Schubladen der Workbench. Wenn Sie alle persönlichen Briefe, die Sie mit einem Textverarbeitungssystem erstellt haben, in eine Schublade legen, alle Rechnungen in eine andere und die kleinen Hilfsprogramme, die Sie nicht täglich benötigen, in eine andere, so wird das Ihre Chancen, schnell einen bestimmten Brief auf der Diskette zu finden, wesentlich erhöhen.

Aber wie Schubladen auf einer Werkbank ihrerseits wieder Kisten und Kästchen enthalten können, die vielleicht jeweils Nägel einer bestimmten Größe enthalten, können auch die

Schubladen auf der elektronischen Workbench weitere Schubladen enthalten. Haben Sie zum Beispiel eine Schublade mit Briefen, so könnten Sie in dieser wiederum Schubladen einrichten, die ältere, neue und noch nicht abgesandte Briefe enthalten. Verwalten Sie sehr viele Briefe elektronisch auf einer Diskette, so dürfte dies wiederum Ihre Chancen, einen bestimmten zu finden, wesentlich verbessern – wie schon zuvor die Entscheidung, Briefe und Rechnungen in getrennte Schubladen zu legen.

Während Sie auf einer richtigen Werkbank aber nicht unendlich viele Schubladen ineinander verschachteln können, ist dies im Prinzip mit den »elektronischen Schubladen« auf der Workbench möglich. Sie können Schubladen in Schubladen in Schubladen in Schubladen in Schubladen ... haben. (Vielleicht kennen Sie ja diese russischen Püppchen, die man öffnen kann, und dann darin wieder ein Püppchen findet, das man öffnen kann und so weiter. Genauso kann man die Schubladen der Amiga-Workbench anwenden.) Wie sinnvoll das in solch extremer Form ist, darüber läßt sich allerdings streiten. Müssen Sie, um etwa einen Brief bearbeiten zu können, zuerst 20 Schubladen öffnen, so ist die Arbeitserleichterung, die Ihnen Schubladen verschaffen können, nicht mehr sehr groß. Wie immer, muß man auch bei der Verwendung von Schubladen das richtige Mittelmaß finden. Eine maximale Schachtelungstiefe von fünf bis sechs Schubladen scheint mir aber durchaus sinnvoll zu sein.

3.2.2 Wie Sie eine Schublade öffnen

Durch den Befehl *Open* im Menü *Workbench* können Schubladen – genau wie auch fast alle anderen Piktogramme – geöffnet werden. Daraufhin öffnet sich ein sogenanntes »Fenster«, das Ihnen den Inhalt der Schublade zeigt. Den Umgang mit Fenstern haben Sie ja bereits in Kapitel 2 kennengelernt.

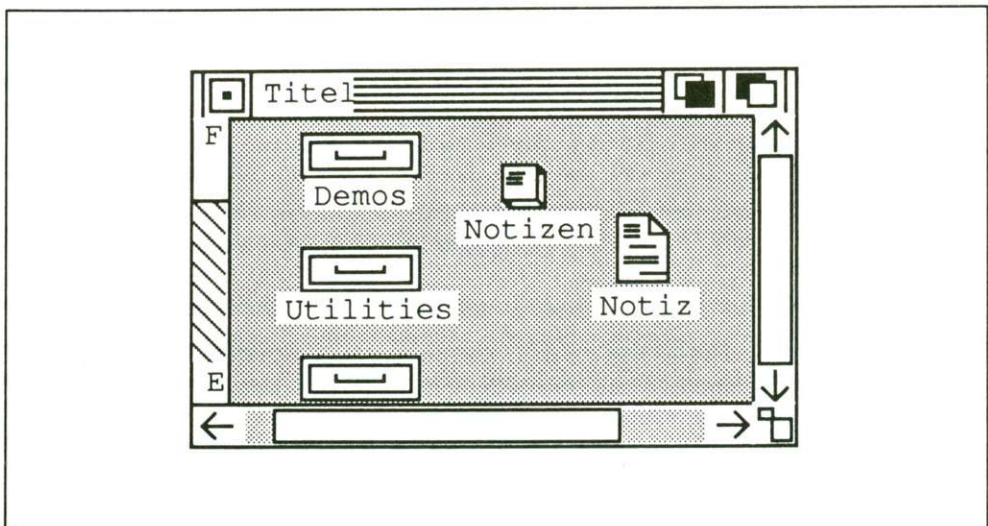


***Bild 3.6:** Das Fenster einer Schublade*

Open hat übrigens auch eine Abkürzung, die so praktisch ist, daß ich die Prophezeiung wage, daß Sie später fast nur noch diese Abkürzung wählen werden und nie mehr den Menü-Befehl *Open*. Klicken Sie einfach zweimal schnell hintereinander mit der linken Maustaste auf ein Disketten-Piktogramm und Sie erzielen dieselbe Wirkung wie beim Auswählen der Diskette und der folgenden Wahl des Befehls *Open* aus dem *Workbench*-Menü. Diesen Vorgang werden wir im folgenden immer *Doppelklick* nennen. Jedes Piktogramm (also auch Disketten-Piktogramme) kann entweder mit dem Menü-Befehl *Open* oder durch einen Doppelklick mit der linken Maustaste darauf geöffnet werden.

3.2.3 Wie Sie Schubladen wieder schließen

Ein Schubladenfenster können Sie jederzeit schließen, indem Sie das Piktogramm der Schublade anklicken und dann den Befehl *Close* aus dem Menü *Workbench* auswählen. Falls Sie sich den »Weg ins Menü« sparen wollen, können Sie auch einfach ins Schließ-Gadget, links oben in der Titelleiste des Fensters, klicken. Auch dadurch wird das Fenster wieder geschlossen.

3.2.4 Wie ein Schubladenfenster aussieht

Wir kommen nun zu zwei weiteren sehr wichtigen Eigenschaften von Fenstern, die auch im vorigen Kapitel noch nicht erwähnt wurden. Diese Eigenschaften werden vor allem dann wichtig, wenn Fenster sehr viele andere Objekte enthalten.

Genauer gesagt, handelt es sich um zwei neue Arten von Gadgets, die nun beschrieben werden sollen. Das erste davon ist das »Größen-Gadget«. Es ist das kleine Quadrat unten rechts im Rahmen eines Fensters mit einem kleinen und einem größeren Rechteck darin. Wie der Name schon sagt, können Sie damit die Größe eines Fensters ändern.

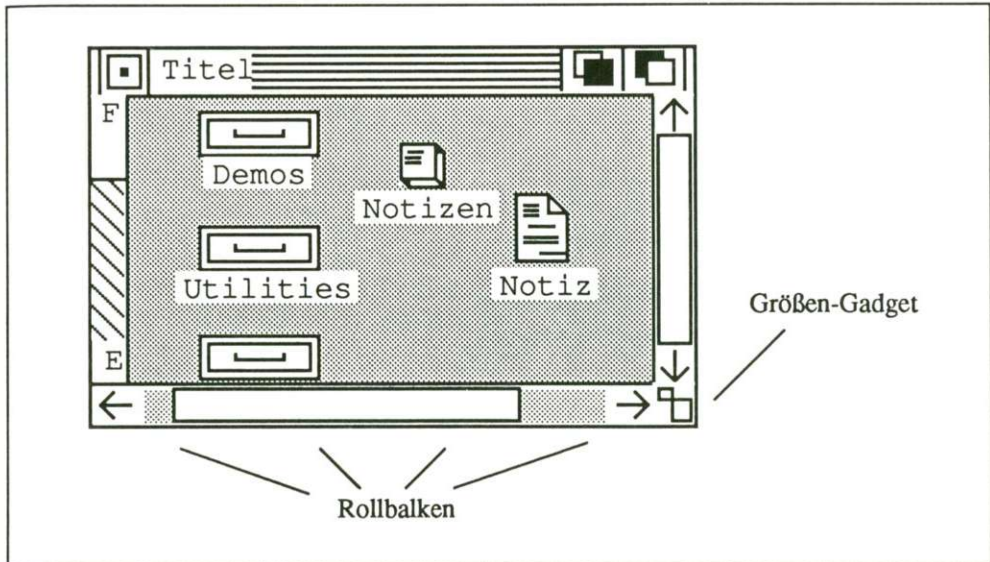


Bild 3.7: Rollbalken und das Größen-Gadget

Wenn Sie das Größen-Gadget mit der Maus ergreifen (das heißt, mit der linken Maustaste darauf klicken und die Taste dann festhalten), so folgt eine rechteckige Umrißlinie bei allen Mausbewegungen. Das sieht ähnlich aus wie beim Verschieben eines Fensters. Diesmal bleibt aber die obere linke Ecke unverändert und nur die untere rechte Ecke folgt (innerhalb gewisser Grenzen) den Mausbewegungen. Sobald die Maustaste losgelassen wird, erhält das Fenster die neue, zuvor durch den flimmernden Rahmen angedeutete Größe.

Enthält ein Fenster nur sehr wenige Objekte, kann man es auf diese Art und Weise wesentlich platzsparender auf dem Bildschirm unterbringen. Liegen andererseits aber sehr viele Objekte darin, so kann man es entsprechend vergrößern. Was aber, wenn die Objekte darin mehr Platz beanspruchen als das Fenster groß ist? In diesem Fall kann innerhalb des Fensters ja nur ein Teil der Fläche gezeigt werden, die eigentlich nötig wäre, um den ganzen Inhalt zu zeigen. Für solche Fälle gibt es die sogenannten »Rollbalken«.

3.2.5 Was Rollbalken sind und wie sie funktionieren

Um den Umgang mit Rollbalken zu üben, öffnen Sie einfach ein Fenster und machen es dann mit dem Größen-Gadget so klein, daß einige der zuvor darin sichtbaren Objekte verschwinden.

Diese Objekte existieren natürlich nach wie vor. Sie sind nur vorübergehend nicht sichtbar. Man kann sich dies veranschaulichen, indem man sich vorstellt, daß alle Objekte, die innerhalb eines Fensters liegen, sich auf einer sehr großen Fläche verteilen, die Platz für (fast) beliebig viele Objekte bietet. Im Fenster selbst kann aber nur ein (eventuell recht kleiner) Ausschnitt

dieser Fläche gezeigt werden. Mit den Rollbalken kann man diesen Ausschnitt, der im Fenster sichtbar ist, auf verschiedene Arten ändern.

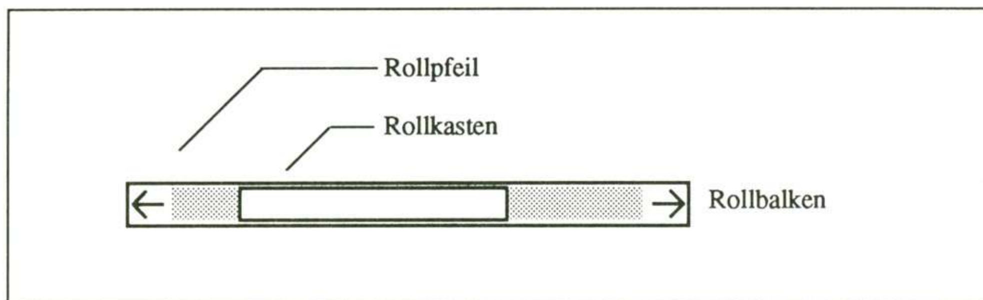


Bild 3.8: *Anatomie eines Rollbalkens*

Was zuerst auffällt, wenn man ein Fenster kleiner macht »als nötig wäre«, sind die beiden Streifen, die Rollbalken am rechten und unteren Rand des Fensters, die ihr Aussehen ändern. Sie tragen an den beiden Enden zwei Pfeile, die in entgegengesetzte Richtungen zeigen. Die Fläche zwischen den Pfeilen ist leer und weiß, solange das Fenster »groß genug« ist. Sobald es aber zu klein wird, wird das weiße Rechteck ebenfalls kleiner und gibt eine »dahinterliegende« blaue Fläche frei.

Diese blaue Fläche symbolisiert das Gesamtbild, von dem ein Teil innerhalb des Fensters gezeigt wird; das weiße Rechteck entspricht dem gerade sichtbaren Teil davon. Das weiße Rechteck nennt man auch *Rollkasten* (übrigens ebenfalls ein Gadget). Wenn der sichtbare Auschnitt eines Fensters kleiner wird, verkleinert sich der Rollkasten entsprechend. Dies ist der Grund dafür, daß der Rollkasten bei einem Fenster, in dem das gesamte Bild sichtbar ist, kaum auffällt. Er füllt dann nämlich den gesamten Raum zwischen den beiden Pfeilen an, weil ja auch das gesamte Bild zu sehen ist.

Die beiden Pfeile an den Enden jedes Rollbalkens sind die »Rollpfeile«. Klickt man auf einen davon, so bewirkt dies, daß der im Fenster sichtbare Ausschnitt des Gesamtbildes ein Stück in die vom Pfeil angedeutete Richtung »rutscht«. Hält man bei diesem Mausklick die Shift-Taste fest, so rückt der Ausschnitt nur um einen Punkt weiter, ansonsten ungefähr um eine halbe Fensterbreite beziehungsweise -höhe. Bei diesem Vorgang rückt auch der Rollkasten in die angegebene Richtung, um anzuzeigen, daß jetzt ein anderer Ausschnitt sichtbar ist.

Der Rollkasten in einem Rollbalken kann aber nicht nur zur Anzeige, sondern auch zur direkten Verschiebung des sichtbaren Ausschnitts verwendet werden. Er läßt sich wie ein Piktogramm mit der Maus ergreifen und innerhalb des Rollbalkens (also nur horizontal oder nur vertikal) verschieben. Läßt man ihn (beziehungsweise die Maustaste) los, »springt« der im Fenster sichtbare Ausschnitt zur entsprechenden Stelle im Gesamtbild.

Die Rollbalken am rechten und unteren Rand eines Fensters verschieben den Ausschnitt vertikal und horizontal, verhalten sich aber ansonsten vollkommen gleich. Zusammen erlauben sie es, den sichtbaren Ausschnitt über einen beliebigen Bereich der »darunterliegenden« Fläche zu verschieben.

3.2.6 Wie Sie Schubladen bewegen

Nach diesem kurzen Exkurs über das Thema Rollbalken nun wieder zurück zu den Schubladen: Sie können Schubladen, genau wie Disketten-Piktogramme auch, mit der Maus bewegen. Die Bewegung von Piktogrammen – nicht nur von Schubladen – erhält durch Schubladen allerdings eine neue Qualität. Wenn Sie ein Objekt (Piktogramm) bewegen und die Maustaste loslassen, während sich das bewegte Piktogramm über dem Piktogramm oder dem Fenster einer Schublade befindet, so wird das Objekt in diese Schublade gelegt. Auf diese Weise kann man Objekte aus einer Schublade (oder aus dem Diskettenfenster) in eine andere Schublade legen.

Dieses Umlegen von Objekten von einer Schublade in eine andere hat allerdings noch eine Besonderheit, wenn sich die beteiligten Schubladen auf verschiedenen Disketten befinden. Liegen beide an einem solchen Vorgang beteiligten Schubladen auf ein und derselben Diskette, läuft alles so ab wie oben beschrieben. Das Objekt verschwindet aus der einen Schublade und erscheint in der anderen.

Befindet sich die zweite Schublade aber auf einer anderen Diskette, so bleibt das verlegte Objekt in der alten Schublade, und in der Schublade auf der zweiten Diskette erscheint eine Kopie des Original-Objektes. Das Verlegen eines Objektes in eine Schublade auf einer zweiten Diskette ist also eine Möglichkeit, eine Kopie dieses Objektes auf einer anderen Diskette anzulegen.

Beim Verlegen von Objekten gelten übrigens auch Disketten als »eine Art« Schubladen! Man kann deshalb auch Objekte genauso in ein Disketten-Piktogramm oder in ein Disketten-Fenster wie in eine Schublade legen.

3.2.7 Wie Sie Schubladen duplizieren

Mit dem Befehl *Duplicate* können Sie ganze Schubladen mit allem, was sich darin befindet, duplizieren. Selektieren Sie dazu mit der Maus das Schubladen-Piktogramm und wählen Sie dann *Duplicate* aus dem *Workbench*-Menü. Daraufhin erscheint eine neue Schublade schräg unterhalb der alten auf dem Bildschirm. Diese neue Schublade erscheint danach neben der alten und erhält zunächst einmal den Namen *copy of <alter Name>* (den man natürlich mit *Rename* ändern kann).

Dieser Befehl kann allerdings aus zwei verschiedenen Gründen fehlschlagen. Falls die Diskette, auf der sich die Schublade befindet schreibgeschützt war, erscheint ein neues Fenster mit einer Fehlermeldung darin auf dem Bildschirm. Wie Sie damit verfahren, wurde bereits weiter oben (im Abschnitt *Wie Fehlermeldungen aussehen*) beschrieben. Aber selbst wenn die Diskette nicht schreibgeschützt ist, kann eine Fehlermeldung auftauchen.

Wenn Sie eine Schublade duplizieren, wird ja nicht nur die Schublade selbst, sondern auch ihr gesamter Inhalt dupliziert. Von jedem Objekt, das sich in der alten Schublade befindet, gibt es danach zwei Versionen auf der Diskette: eine in der alten und eine in der neuen Schublade. Falls auf der Diskette aber nicht mehr genügend Platz für all die neuen Objekte ist, erscheint wieder eine Fehlermeldung, die besagt, daß die Diskette voll ist (engl. *Volume <name> is full*).

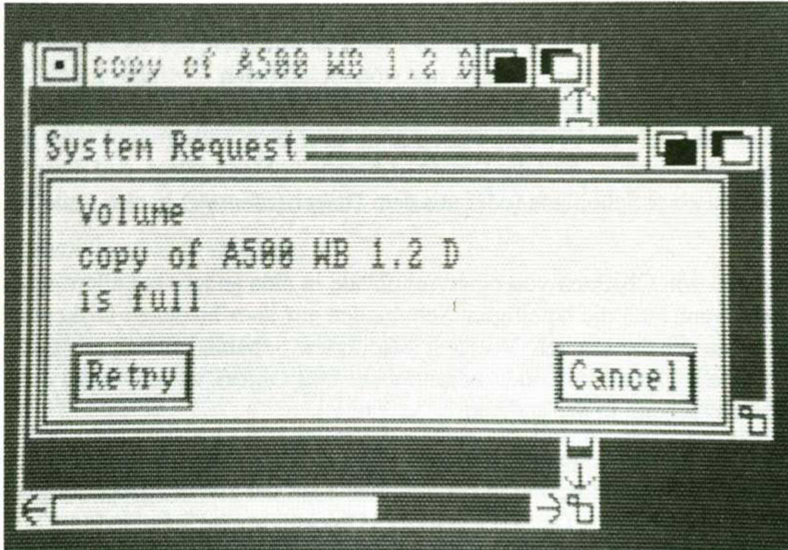


***Bild 3.9:** Die Fehlermeldung, die beim »Überlaufen« einer Diskette erscheint*

Danach liegt zwar eine neue Schublade auf der Workbench. Dies ist aber keine vollständige Kopie der Originalschublade. Sie sollten diese unvollständige Kopie sicherheitshalber löschen (siehe unten). Wenn Sie das nicht tun, kann das später zu Fehlern führen.

3.2.8 Wie das Info-Fenster einer Schublade aussieht

Der Befehl *Info* liefert Ihnen detailliertere Informationen über eine Schublade. Selektieren Sie dazu die Schublade, die Sie interessiert, mit der Maus und wählen Sie dann den Befehl *Info* aus dem Menü *Workbench*. Das Fenster, in dem Ihnen diese Informationen dargeboten werden, sieht ein klein wenig anders aus als das Info-Fenster einer Diskette.

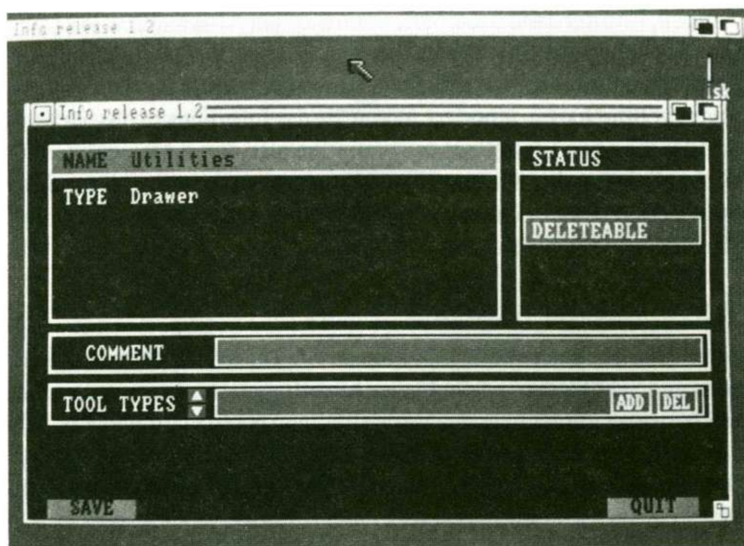


Bild 3.10: Das Info-Fenster einer Schublade

Das Info-Fenster zeigt für eine Schublade im Gegensatz zu einer Diskette nicht den Speicherbedarf der Objekte in der Schublade an. Dafür haben Sie jedoch die Möglichkeit, einen zusätzlichen Kommentar zu vermerken, der Ihnen später hilft, wenn Sie einmal nicht mehr genau wissen, was Sie nun denn in die Schublade gelegt haben. Klicken Sie zum Anbringen eines Kommentars einfach in das blaue Rechteck hinter COMMENT (engl. für *Kommentar*). Sie können in diesem Feld dann einen beliebigen Text eintippen und müssen die Eingabe mit der [Return]-Taste beenden. Zur Fehlerkorrektur innerhalb des Rechtecks stehen Ihnen übrigens genau dieselben Möglichkeiten zur Verfügung wie beim Befehl *Rename* (siehe oben).

Sie können eine Schublade auch vor unbeabsichtigtem Löschen schützen. Klicken Sie dazu einfach im Rechteck mit dem Titel STATUS das Feld an, in dem das Wort DELETEABLE (engl. für *löschbar*) steht. An dieser Stelle erscheint nun NOT DELETEABLE (engl. für *nicht löschbar*). Versuchen Sie später, diese Schublade zu löschen, so erhalten Sie nur eine Fehlermeldung – die Schublade selbst bleibt erhalten. Bevor sie wirklich gelöscht werden kann, muß mit dem *Info*-Befehl dann zunächst der »Schalter« unter STATUS wieder auf DELETEABLE umgelegt werden. Dieser Schutz für Schubladen ist allerdings nicht allzu sicher. Er verhindert zum Beispiel nicht ein Löschen der gesamten Diskette (und damit natürlich auch dieser Schublade) mit dem *Initialize*-Befehl. Wirklich wichtige Disketten sollten Sie deshalb immer mit dem Schreibschutz der Disketten sichern.

Das Info-Fenster verschwindet (wird geschlossen), wenn Sie in das Schließ-Gadget den Knopf QUIT oder den Knopf SAVE klicken. Die Änderungen im Info-Fenster, die gerade

beschrieben wurden, sind nur dann dauerhaft, wenn Sie zum Abschluß mit der Maus den Knopf SAVE drücken. Schließen Sie das Fenster mit dem Schließ-Gadget oder indem Sie auf den Knopf QUIT unten rechts in der Ecke drücken, so werden sämtliche Änderungen, die Sie im Info-Fenster durchgeführt haben, ignoriert. Wenn Sie das Info-Fenster das nächste Mal öffnen, erhalten Sie wieder die alten Einstellungen. Wenn Sie also den Kommentar einer Schublade geändert oder den Schalter im Kasten STATUS umgelegt haben, sollten Sie das Fenster unbedingt mit SAVE verlassen. Nur dann werden Ihre Änderungen auch wirklich berücksichtigt und dauerhaft gespeichert.

3.2.9 Wie Sie neue Schubladen erzeugen

Anders als bei einer wirklichen Werkbank können Sie der Workbench des Amiga jederzeit neue Schubladen und Schubladen in Schubladen hinzufügen. Das Erzeugen neuer Schubladen läuft dabei üblicherweise immer über das Vervielfältigen bereits vorhandener Schubladen ab. Dies ist ganz einfach. Wie bereits oben erwähnt, können Sie Schubladen duplizieren, indem Sie sie auswählen und dann den Befehl *Duplicate* aus dem *Workbench*-Menü aufrufen. Bei Schubladen bewirkt dieser Befehl das Anlegen einer Kopie der ursprünglichen Schublade und ihres gesamten Inhalts. Auf diese Art und Weise entsteht natürlich eine neue Schublade – eben die Kopie.

Benötigen Sie aber eine neue leere Schublade, um darin zum Beispiel einen Teil der Objekte aus einer anderen Schublade unterzubringen, so müssen Sie auch eine leere Schublade duplizieren. Auf der Standard-Workbench-Diskette ist deshalb eine leere Schublade untergebracht worden, die genau für diesen Zweck gedacht ist. (Sie trägt den Namen *Empty*; zu deutsch *leer*.)

Wenn Sie in irgendeiner Schublade eine neue Unterschublade unterbringen wollen, müssen Sie deshalb diese leere Schublade im Disketten-Fenster der Workbench-Diskette auswählen und duplizieren. Mit dem Befehl *Rename* (siehe oben) können Sie der Kopie danach den gewünschten Namen geben. Zuletzt können Sie sie dann in die Schublade legen, in der Sie eine neue Schublade benötigen. Sie können die Schublade *copy of Empty* aber natürlich auch zuerst dorthin legen, wo Sie sie benötigen und dann umbenennen.

3.2.10 Eine besondere Schublade: der Mülleimer

Ein Symbol, das Ihnen bestimmt sofort aufgefallen ist, nachdem Sie das erste Mal ein Disketten-Piktogramm geöffnet haben, ist der Mülleimer (engl. *Trashcan*). Der Mülleimer dient zum Löschen anderer Objekte. Er verhält sich dabei zum Teil wie eine »Schublade«.

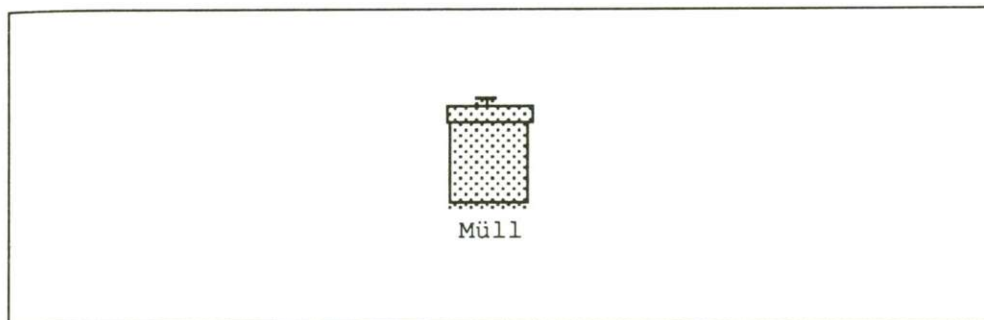


Bild 3.11: Das Mülleimer-Piktogramm

Wollen Sie ein beliebiges Objekt auf der Workbench löschen, so nehmen Sie es einfach und legen es in den Mülleimer auf derselben Diskette. Das geht genauso wie das Verlegen eines Objektes in eine andere Schublade. Man ergreift das Objekt mit der Maus, bewegt es über das Mülleimer-Piktogramm und läßt die Maustaste los.

Wenn man ein Objekt in den Mülleimer gelegt hat, ist es noch nicht sofort verloren. Wie »in Wirklichkeit« liegt es zunächst einmal im Mülleimer und kann wieder daraus hervorgeholt werden – solange die Müllabfuhr noch nicht da war. Hierzu kann man den Mülleimer öffnen wie jede andere Schublade auch. Daraufhin erscheint ein Schubladenfenster, in dem alle Objekte auftauchen, die man bisher in den Mülleimer geworfen hat. Haben Sie es sich anders überlegt und wollen Sie eines davon nicht löschen, so können Sie es mit der Maus greifen und in eine andere Schublade legen.

Erst wenn die Müllabfuhr gekommen ist, sind die Objekte, die in den Mülleimer geworfen wurden, unwiederbringlich verloren. Auf dem Amiga können Sie selbst bestimmen, wann die Müllabfuhr kommt. Wenn Sie ganz sicher sind, die Objekte im Mülleimer nicht mehr zu benötigen, selektieren Sie das Mülleimer-Piktogramm und wählen danach *Empty Trash* aus dem *Disk*-Menü. (Der Befehl ist nur dann möglich, wenn Sie wirklich zuvor ein Mülleimer-Piktogramm selektiert haben; sonst erscheint er in Geisterschrift. Der selektierte Mülleimer läßt sich an seiner blauweißen Färbung – »normal« ist er orangeschwarz – erkennen.) Daraufhin werden die Objekte im Mülleimer wirklich von der Diskette gelöscht. Enthält der Mülleimer eines oder mehrere Objekte, die mit dem *Info*-Befehl vor Löschung geschützt wurden, so ergibt *Empty Trash* allerdings eine Fehlermeldung und die geschützten Objekte werden auch nicht entfernt. Es kann aber durchaus sein, daß zu dem Zeitpunkt, wo der Amiga diesen Fehler bemerkt hat, bereits Objekte, die im Mülleimer lagen und nicht geschützt waren, gelöscht wurden!

Mülleimer sind auch insofern besondere Schubladen, weil man sie nicht aus dem Diskettenfenster entfernen kann. Sie können weder in eine andere Schublade noch in ein anderes Diskettenfenster gelegt werden. Auch können Mülleimer nicht mit dem Befehl *Discard* (siehe unten) gelöscht werden. Besitzt eine Diskette erst einmal ein Mülleimer-Piktogramm, so kann dieses nicht wieder entfernt werden! (Zumindest nicht von der

Workbench aus. Vom CLI aus, das in einer Kapitelfolge erläutert wird, kann auch der Mülleimer gelöscht werden.)

3.3 Disketten

Disketten sind nach den Schubladen die zweitwichtigste Gruppe von Piktogrammen. In vieler Hinsicht können Disketten genauso behandelt werden wie Schubladen. Sie können zum Beispiel auf dieselbe Weise geöffnet und geschlossen, umbenannt und dupliziert werden. Noch wichtiger ist aber, daß Diskettenfenster und Disketten-Piktogramme sich ähnlich verhalten wie Schubladenfenster und Schubladen-Piktogramme.

Ergreift man zum Beispiel ein Objekt mit der Maus und läßt es über einem Disketten-Piktogramm fallen (indem man dort die Maustaste löst), so verschwindet das Objekt an seiner alten Position und erscheint im Fenster dieser Diskette (sofern dieses geöffnet ist). Man kann Objekte auch direkt in das Diskettenfenster legen. Befand sich das »alte Objekt« vor seiner Verlegung in einer Schublade, die sich auf einer anderen Diskette befand, so findet auch beim Verlegen in ein Diskettenfenster oder -Piktogramm ein Kopiervorgang statt. Auf der zweiten Diskette erscheint eine Kopie des Objektes. Das alte Objekt selbst verbleibt in seiner »alten« Schublade.

In einer Hinsicht unterscheiden sich Disketten aber von Schubladen. Disketten können zwar Schubladen enthalten, umgekehrt gilt dies jedoch nicht. Versucht man, ein Disketten-Piktogramm in ein anderes Fenster oder auf ein Schubladen-Piktogramm zu legen, so erscheint eine Fehlermeldung. Disketten können allerdings in andere Disketten-Piktogramme gelegt werden, wobei der Inhalt dieser zweiten Diskette komplett durch den der bewegten Diskette ersetzt wird. Diesen Vorgang haben Sie ja bereits im Kapitel 2 kennengelernt.

3.3.1 Wie ein Diskettenfenster aussieht

Wenn Sie einen Doppelklick (mit der linken Maustaste) auf ein Disketten-Piktogramm machen, öffnet sich am Bildschirm ein neues Fenster, in dem der Disketteninhalt dargestellt wird.

Den Aufbau und die möglichen Operationen mit einem Diskettenfenster haben Sie zum großen Teil ebenfalls schon im Kapitel 2 und im Abschnitt über Schubladen kennengelernt. Hauptunterschied zwischen einem Disketten- und einem Schubladenfenster ist die *Füllanzeige*. Dies ist am linken Rand des Fensters, gegenüber dem vertikalen Rollbalken angebracht. Sie besteht aus einem schmalen senkrechten Balken, dessen unterer Teil orange und dessen oberer schwarz ist. Am unteren Ende des Balkens steht ein E für *empty* (engl. für *leer*), am oberen Ende ein F für *full* (engl. für *voll*). Der orangefarbene Teil kennzeichnet den Teil der Diskette, der bereits gefüllt ist, der schwarze den noch freien Teil. Nimmt der orangefarbene Balken also die Hälfte des Balkens ein, ist die Diskette halb voll – oder halb leer; wie Sie wollen. Ist der ganze Balken orange, ist die Diskette voll und kann keine neuen Daten mehr aufnehmen. Sie haben mit diesem Balken immer einen ungefähren Überblick über den »Füllgrad« einer Diskette.

Benötigen Sie genauere Daten, so können Sie dies mit dem *Info*-Befehl herausfinden, der im folgenden Abschnitt erläutert wird.

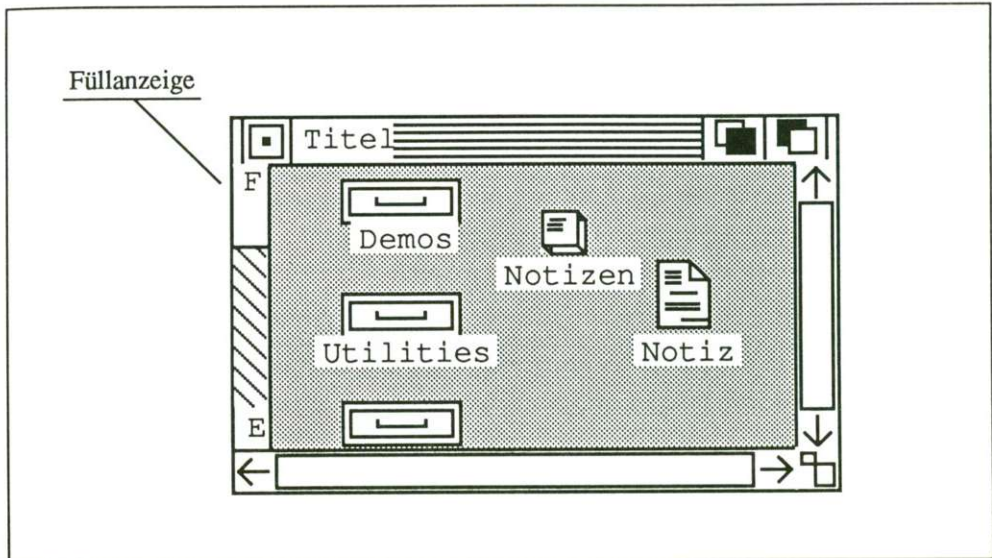


Bild 3.12: *Die Disketten-Füllanzeige*

3.3.2 Wie das Info-Fenster einer Diskette aussieht

Wenn Sie mehr über eine Diskette wissen wollen, als nur ihren Namen, so müssen Sie, wie üblich, das Disketten-Piktogramm auswählen und dann den Befehl *Info* aus dem Menü *Workbench* wählen. Das Info-Fenster zeigt in der oberen linken Ecke den Typ (*TYPE Disk*) und, unter der Typangabe, einen Überblick über die exakte Platzausnutzung der Diskette. Hinter *Number of Blocks* erfahren Sie, in wieviele »physikalische Blocks« die Diskette aufgeteilt ist (bei 3,5-Zoll-Disketten ist die Zahl immer gleich 1758). Darunter finden Sie die Angaben darüber, wieviele dieser Blocks belegt (hinter *Number Used*) und wieviele noch frei sind (hinter *Number Free*). Die letzte Angabe schließlich (*Bytes per Block*) sagt Ihnen, wieviele Buchstaben beziehungsweise Bytes in jedem der Blocks Platz haben, von denen oben die Rede war. Bei einer Diskette sind es üblicherweise 512; falls Sie Besitzer einer Festplatte sind, kann hier aber durchaus eine andere Zahl stehen. (Die Diskette kann damit also maximal 1758 Block zu 512 Byte ≈ 900 000 Byte aufnehmen.)

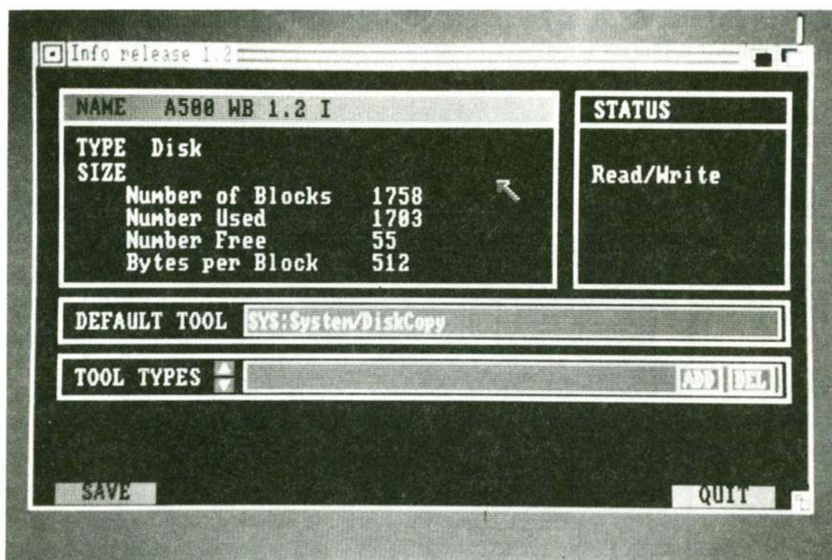


Bild 3.13: Das Info-Fenster einer Diskette

Neben diesem Feld mit den vielen Zahlen befindet sich das Feld STATUS. Falls Sie die Diskette mit dem kleinen Plastikschieber schreibgeschützt haben, steht darin *Read Only* (engl. für *nur lesen*) ansonsten immer *Read/Write* (engl. für *lesen und schreiben*). Die beiden Felder darunter (*Default Tool* und *Tool Types*) sollen uns vorläufig nicht interessieren. Wie Sie diese Felder bei bestimmten anderen Arten von Piktogrammen sinnvoll verwenden können, werden Sie später noch erfahren.

3.3.3 Wie Sie Disketten kopieren

Disketten können, wie alle anderen Piktogramme, natürlich auch kopiert beziehungsweise dupliziert werden. Hierzu gibt es sogar zwei Wege. Beide sind im Kapitel 2 bereits beschrieben worden. Hier nur noch einmal ein paar kurze Stichworte zur Erinnerung:

Wenn Sie nur ein Diskettenlaufwerk besitzen, sollten Sie das Piktogramm der Diskette selektieren und dann den Befehl *Duplicate* aus dem *Workbench*-Menü wählen, wenn Sie eine Diskette kopieren wollen. Danach erscheint nacheinander eine Reihe von Meldungsfenstern (Requestern), die Sie auffordern, entweder die Originaldiskette (*FROM disk*) oder die Diskette, die die Kopie aufnehmen soll (*DESTINATION disk*), einzulegen. Folgen Sie dieser Aufforderung und drücken Sie dann jeweils auf den Knopf *Continue*.

Wenn Sie mit zwei Diskettenlaufwerken arbeiten, so legen Sie die schreibgeschützte Originaldiskette in ein Laufwerk, und eine neue Diskette, die die Kopie aufnehmen soll, in das zweite Laufwerk. Ergreifen Sie dann das Piktogramm der Originaldiskette mit der Maus und legen Sie es auf das Piktogramm der zweiten Diskette (dieses trägt den Namen *DF1:BAD*,

wenn es sich um eine neue Diskette handelt). Dann erscheint wieder ein Requester, mit den Knöpfen *Cancel* und *Continue*. Wenn Sie sicher sind, alles richtig gemacht zu haben, brauchen Sie nur noch auf *Continue* zu klicken, und der Kopiervorgang beginnt. *Cancel* bricht den Befehl ab.

3.3.4 Wie Sie Disketten löschen und initialisieren

Wenn Sie auf eine der eben beschriebenen Arten eine ganze Diskette kopieren, so kann die Diskette, die die Kopie aufnehmen soll, frisch aus der Packung kommen. Sie brauchen sie nicht weiter vorzubereiten, damit sie Daten aufnehmen kann. Wenn Sie jedoch einzelne Piktogramme (zum Beispiel eine Schublade) auf eine neue Diskette legen wollen, muß die Diskette dazu erst einmal initialisiert werden. Dieser Vorgang teilt die zunächst unstrukturierte magnetische Oberfläche in einzelne Sektoren beziehungsweise Blöcke zu 512 Byte auf, über die ja zum Beispiel auch der *Info*-Befehl Auskunft gibt. Auch wird dabei Platz für das Dateiverzeichnis angelegt, eine Art Landkarte, die den Programmen sagt, wo welche Daten auf der Diskette zu finden sind. Beim Kopieren einer ganzen Diskette wird die Diskette, die die Kopie aufnimmt, auch gleich initialisiert. Dies geschieht automatisch, ohne Ihr Zutun und ohne, daß Sie etwas davon bemerken.

Um eine neue Diskette zu initialisieren, müssen Sie diese also zunächst (mit der Maus) selektieren (sie trägt den Namen *DF1:BAD*, wenn es sich wirklich um eine frische, noch unbenutzte Diskette handelt) und dann den Befehl *Initialize* im Menü *Disk* auswählen. Bevor der Initialisierungs-Vorgang beginnt, werden Sie in einem Requester sicherheitshalber gefragt, ob Sie sich Ihrer Sache sicher sind. Nur wenn Sie in diesem Fenster den Knopf *Continue* anklicken, wird die selektierte Diskette initialisiert und damit gelöscht. Klicken Sie aber auf Knopf *Cancel*, wird der Vorgang abgebrochen.

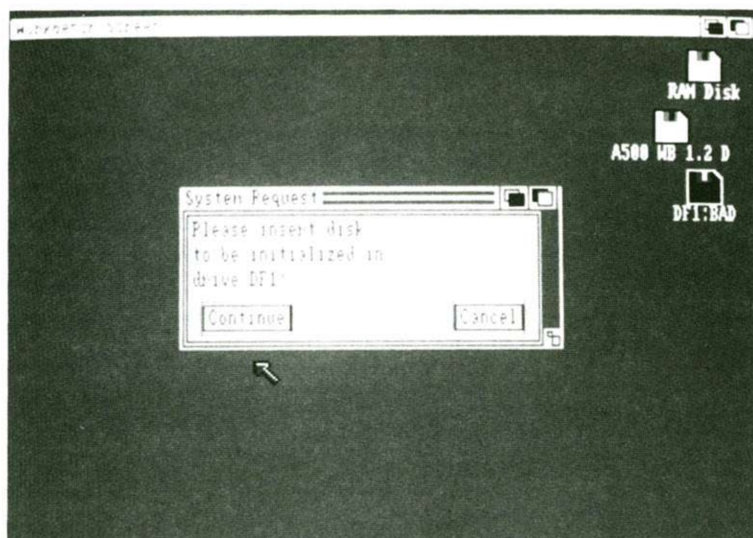


Bild 3.14: Requester zur Bestätigung des Initialisierens einer Diskette

Das Initialisieren kann nicht nur bei neuen Disketten nötig sein. Es ist auch manchmal sinnvoll, bereits verwendete Disketten, deren Inhalt Sie (vielleicht aus Geheimhaltungsgründen) löschen wollen, zu initialisieren. Das Initialisieren zerstört alle Daten, die sich eventuell vorher auf einer Diskette befanden! Achten Sie deshalb beim Initialisieren von schon benutzten Disketten darauf, daß es auch die »richtigen« sind. Sie können sich aber vor Fehlern schützen, indem Sie den Schreibschutz wichtiger Disketten aktivieren. Schreibgeschützte Disketten können nicht initialisiert werden!

3.4 Andere Piktogramme: Werkzeuge und Projekte

Alle Objekte, die wir bis jetzt kennengelernt haben, waren mehr oder weniger Ordnungshilfen für Disketten. Die Piktogramme, die diese repräsentieren, sind Disketten, Schubladen und Mülleimer. Es gibt aber auch noch andere Piktogramme in den Fenstern – zum Beispiel der Workbench-Diskette und in ihren Schubladen. Bis jetzt wurde ja immer sehr abstrakt von »Objekten« gesprochen, womit alle möglichen Piktogramme in einem Schubladenfenster gemeint waren. Disketten, Schubladen und Mülleimer haben wir bereits kennengelernt. Was aber steckt hinter den anderen Piktogrammen auf der Workbench?

3.4.1 Werkzeuge und Projekte

Alle Objekte, die Sie in einem Disketten- oder Schubladenfenster sehen können, und bei denen es sich nicht um Schubladen oder Mülleimer handelt, sind *Werkzeuge* und *Projekte*.

Getreu der Analogie, daß die Bedienung des Amiga abläuft wie die Arbeit auf einer Werkbank (engl. *workbench*), heißen die Programme des Amiga nämlich Werkzeuge (engl. *tools*) und die Dateien, die diese Programme erzeugen oder für ihre Arbeit benötigen, heißen Projekte (engl. *projects*). Ein Brief oder ein kurzes Memo, das Sie mit einem Textverarbeitungsprogramm *Textcraft* geschrieben haben, wäre also ein Projekt, das mit dem Werkzeug *Textcraft* bearbeitet wurde.

Jedes Werkzeug kann als nahezu beliebiges Piktogramm am Bildschirm erscheinen. Meist deutet das Aussehen des Piktogramms aber die Funktion des Programms an. Eine Textverarbeitung wird also vielleicht wie eine kleine Schreibmaschine aussehen und ein Grafikprogramm wie eine Palette mit Pinsel.

Zwischen Werkzeugen und den Projekten, die man damit bearbeitet, besteht ein enger Zusammenhang. Meist kann ein Projekt nur mit einem ganz bestimmten Werkzeug bearbeitet werden. Eine Textverarbeitung kann mit Grafiken nichts anfangen und umgekehrt (Ausnahmen kommen allerdings vor). Dieser Zusammenhang wird meist dadurch angedeutet, daß die Piktogramme von Projekten logisch zu den Werkzeug-Piktogrammen passen. Texte werden also zum Beispiel wie ein Blatt Papier mit einem Eselsohr und ein paar angedeuteten Buchstaben aussehen und von der Aufmachung (zum Beispiel den Farben) her dem Textverarbeitungsprogramm ähnlich sehen.

Wenn Sie jetzt gerne einmal ein solches Werkzeug sehen wollen, öffnen Sie einfach die Workbench-Diskette. Warten Sie einen Moment, bis das Diskettenlaufwerk zur Ruhe gekommen ist. Oben im Fenster sind nun zwei Objekte zu sehen, die weder wie Schubladen noch wie Disketten oder Mülleimer aussehen. Dies sind die beiden Programme (Werkzeuge) *Clock*, *Preferences* und eventuell *CLI*. *Clock* (engl. für *Uhr*) zeigt Ihnen zum Beispiel – wie der Name schon sagt – am Bildschirm eine Uhr.

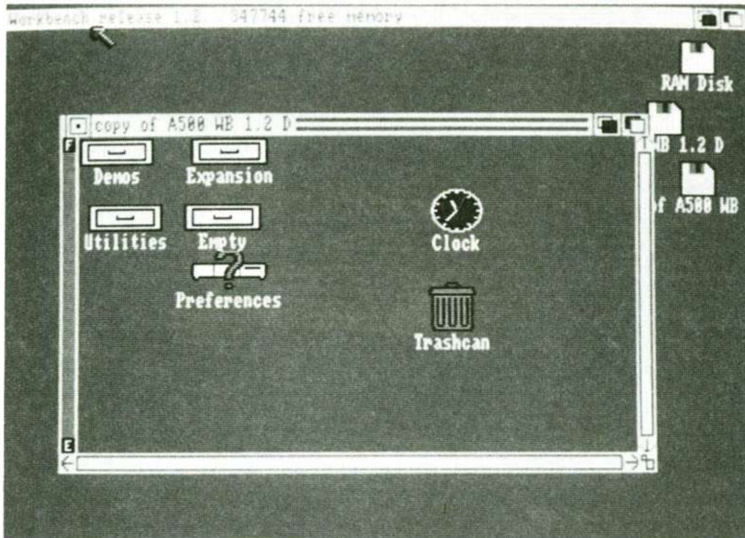


Bild 3.15: *Die Werkzeuge Clock und Preferences*

Damit sind die wesentlichen Eigenschaften von Werkzeugen und Projekten aber auch schon geklärt. Im Rest dieses Kapitels sollen Werkzeuge und Projekte aber nur aus der Sicht der Workbench betrachtet werden. Das folgende Kapitel wird sich näher mit diesen Objekten beschäftigen.

3.4.2 Operationen für Werkzeuge und Projekte

Auf der Workbench können Werkzeuge und Projekte wie alle anderen Piktogramme behandelt werden. Die meisten Befehle und Operationen funktionieren wie bei Schubladen. Werkzeuge und Projekte können zum Beispiel mit der Maus ergriffen und bewegt werden. Sie können sie in Schubladen legen, indem Sie sie in das Schubladen-Piktogramm oder in ein Schubladenfenster legen. Werden sie in eine Schublade, die sich auf einer anderen Diskette befindet, oder direkt in das entsprechende Diskettenfenster gelegt, so werden auch Werkzeuge und Projekte auf diese andere Diskette kopiert, wie es bei einer Schublade der Fall wäre.

Aus dem Menü *Disk* können keine Befehle angewendet werden, wenn ein Werkzeug oder Projekt selektiert ist. Dafür sind fast alle Befehle des Menüs *Workbench* anwendbar.

Der Befehl *Open* wird ausführlich im nächsten Kapitel erläutert. (Sie sollten ihn deshalb vorläufig nicht auf Projekt- oder Werkzeug-Piktogramme anwenden und auch keinen Doppelklick auf ein solches Piktogramm machen!)

Der Befehl *Close* kann nicht auf Werkzeuge oder Projekte angewendet werden. Mit dem Befehl *Duplicate* kann (auf derselben Diskette) eine Kopie eines Werkzeugs oder Projektes

angelegt werden, die zunächst den Namen *copy of <alter Name>* trägt. Diesen Namen kann man danach mit dem Befehl *Rename* beliebig ändern.

Mit dem Befehl *Info* bekommen Sie detailliertere Informationen über ein Werkzeug oder Projekt in einem neuen Fenster angezeigt, das ganz ähnlich aussieht wie das Info-Fenster einer Schublade. Näheres über die Möglichkeiten, die Ihnen im Info-Fenster eines Projekts oder eines Werkzeugs zur Verfügung stehen, erfahren Sie im nächsten Kapitel.

3.5 Weitere Befehle und Operationen

Die meisten Befehle der Workbench haben Sie nun – in diesem oder im Kapitel 2 – kennengelernt. Einige Befehle sind bisher aber unerwähnt geblieben, wie Ihnen vielleicht aufgefallen sein wird, wenn Sie die Menüs einmal aufmerksam durchblättert haben. Diese Befehle und einige zusätzliche Tips und Handgriffe werden Sie in den folgenden Abschnitten kennenlernen.

3.5.1 Wie Sie mit dem Discard-Befehl Objekte löschen

Wenn Sie täglich mit dem Amiga arbeiten, wird es mit Sicherheit gelegentlich nötig werden, auf Ihren Disketten »aufzuräumen«. Damit ist natürlich unter anderem auch gemeint, daß alle Objekte ordentlich in Schubladen untergebracht werden, damit man sie später auch wiederfindet. Ab und zu werden Sie aber auch Objekte oder Schubladen, die durch eine Reorganisation überflüssig geworden sind, löschen müssen. Die sicherste Methode, ein Objekt zu löschen, besteht darin, es zunächst einmal in den Mülleimer zu legen und später, wenn der Platz auf der Diskette knapp wird, den Mülleimer leeren zu lassen. Der Vorteil des Mülleimers ist, daß Sie es sich auch »noch einmal anders überlegen können«. Objekte, die im Mülleimer sind, sind ja auch in Wirklichkeit noch nicht zerstört oder verloren, sondern können wieder hervorgeholt werden. Diese Löschmethode ist andererseits aber relativ umständlich und langwierig, wenn Sie etwas Bestimmtes **sofort** löschen wollen, und sich Ihrer Sache sehr sicher sind. Für solche Fälle gibt es den Befehl *Discard*.

Selektieren Sie dazu das Objekt, das Sie löschen wollen, durch Anklicken mit der Maus und wählen Sie dann den Befehl *Discard* aus dem Menü *Workbench* aus. Daraufhin erscheint ein Requester, der Sie über die Konsequenzen (unwiederbringlicher Verlust der Schublade und aller Objekte darin) informiert und Ihnen noch einmal die Chance gibt, es sich anders zu überlegen.

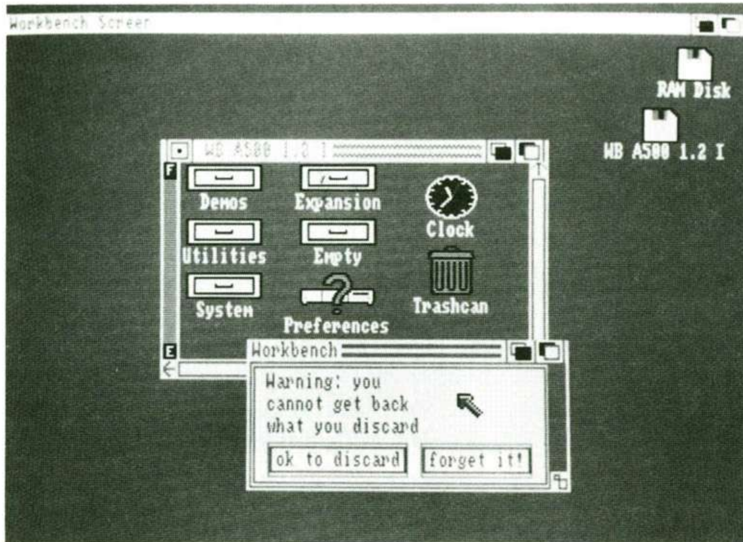


Bild 3.16: *Der Requester zur Bestätigung des Discard-Befehls*

Dieses Fenster besitzt am unteren Rand zwei Knöpfe, *ok to discard* (engl. für *ich bin damit einverstanden, dieses Objekt zu löschen*) und *forget it!* (engl. *vergiß es*). Drückt man (mit Mauszeiger und Maustaste) auf *ok to discard*, so wird der Löschvorgang durchgeführt, und danach gibt es wirklich keine Möglichkeit mehr, das selektierte Objekt zu retten. Drückt man aber auf den Knopf *forget it!* , wird der Vorgang abgebrochen und es geschieht nichts weiter.

Mit *Discard* kann man fast alle Objekte auf einer Diskette löschen. Das Löschen von Schubladen ist aber besonders gefährlich, da man oft nicht mehr genau weiß, was diese enthalten. Vor dem Löschen einer Schublade (egal ob mit *Discard* oder über den Mülleimer) sollte man diese öffnen und sich vergewissern, daß nur Objekte in ihr liegen, die nicht mehr benötigt werden! Schubladen mit wichtigem Inhalt kann man auch (über den *Info*-Befehl) vor versehentlichem Löschen schützen. Dies ist aber kein sehr perfekter Schutz, da er nicht vor einem Öffnen der Schublade und dem Löschen der einzelnen Objekte darin schützt. Im allgemeinen ist es aber doch wohl so, daß nicht die Schublade, sondern ihr Inhalt wichtig ist.

Den Mülleimer sollten Sie aber der Verwendung des Befehls *Discard* im Zweifelsfall vorziehen. Legen Sie alle Objekte, die Sie löschen wollen, zunächst in den Mülleimer und leeren Sie diesen erst, wenn Sie dringend mehr Platz auf der entsprechenden Diskette benötigen! Dies läßt Ihnen die Chance, die so »fortgeworfenen« Objekte bei einem Irrtum wieder zu holen.

3.5.2 Wie Sie eine Schublade aufräumen

Die Objekte, die in einem Schubladen- oder Diskettenfenster gezeigt werden, können Sie mit der Maus an jede beliebige Stelle schieben. Dadurch kann der Fensterinhalt manchmal etwas »unaufgeräumt« erscheinen. Falls Sie ein bißchen Ordnung im Diskettenfenster schaffen wollen, können Sie den Befehl *Clean up* (engl. für *aufräumen*) verwenden. Wählen Sie dazu das Disketten- oder Schubladen-Piktogramm aus, das zu dem Fenster gehört (es reicht nicht, das Fenster anzuklicken!) und wählen Sie dann *Clean up* aus dem Menü *Special*. Die Piktogramme werden dann vom Amiga selbst in »Reih und Glied« gelegt.

Die Ergebnisse des *Clean up*-Befehls sind allerdings nicht immer so, wie man sich die Ordnung auf seiner Diskette vorstellt. Dies liegt unter anderem daran, daß die Piktogramme verschieden groß sein können. Sind Sie mit dem Ergebnis des Aufräumens nicht zufrieden, bleibt Ihnen nichts anderes übrig, als die Objekte selbst »von Hand« so zu legen, wie Sie es für richtig halten.

3.5.3 Wie Sie einen Schnappschuß machen

Wenn Sie ein Piktogramm zu einer neuen Position legen, ein Fenster verschieben oder vergrößern, so »halten« alle diese Änderungen nur bis zum Abschalten des Amiga oder bis zum nächsten Neustart. Wenn Sie wollen, daß solche Änderungen auch einen Neustart überstehen, müssen Sie einen Schnappschuß davon machen, der diese Änderungen auf der Diskette abspeichert. Legen Sie dazu ein Piktogramm mit der Maus an die gewünschte Position und wählen dann den Befehl *Snapshot* (engl. für *Schnappschuß*) aus dem Menü *Special*. Dieser Befehl speichert die aktuelle Position des Piktogramms auf der Diskette ab. Auch beim nächsten Start des Amiga erscheint das Piktogramm dann an dieser neuen Position. Ein Schnappschuß ist natürlich nur möglich, wenn die Diskette nicht schreibgeschützt ist. Versuchen Sie einen Schnappschuß auf einer schreibgeschützten Diskette zu machen, so ergibt dies eine Fehlermeldung.

Wenn es sich bei dem Piktogramm, von dem Sie einen Schnappschuß machen, um eine Diskette oder eine Schublade handelt und deren Fenster gerade offen ist, wird nicht nur die Position des Piktogramms, sondern auch die Größe und Position des Fensters mit abgespeichert. Immer wenn Sie einen Schnappschuß von Schubladen oder Disketten anwenden, sollten Sie deshalb darauf achten, daß Ihnen auch die Position und Größe, die das Fenster gerade hat, gefallen.

3.5.4 Wie Sie mehr als ein Piktogramm mit einem Befehl bearbeiten

Wenn bislang in diesem Buch ein Befehl oder eine Operation vorgestellt wurde, so wurde immer nur beschrieben, wie Sie diesen Befehl auf genau **ein** Piktogramm anwenden können. Wenn Sie mehrere Piktogramme zu bearbeiten haben, zum Beispiel zwei Schubladen auf einmal kopieren wollen oder einen Schnappschuß von drei Werkzeugen machen wollen, kann das wiederholte Aufrufen desselben Befehls lästig werden. Deswegen gibt es die *erweiterte Selektion*, durch die es möglich ist, mit einem Befehl gleichzeitig mehrere Piktogramme zu bearbeiten.

Klicken Sie dazu zunächst das erste Piktogramm an, das Sie bearbeiten wollen. Es zeigt die Selektion durch eine Farbänderung an. Drücken Sie dann eine der beiden Shift-Tasten rechts

und links unten auf der Tastatur (die Umschalttasten, mit denen Sie sonst zwischen Klein- und Großbuchstaben wechseln) und halten sie fest. Klicken Sie dann bei gedrückter Shift-Taste das nächste Piktogramm an. Auch dieses zeigt nun durch Farbänderung an, daß es selektiert wurde, während das erste selektiert bleibt. (Normalerweise führt die Selektion eines Piktogramms dazu, daß das zuletzt ausgewählte Piktogramm deselektiert wird.) Fahren Sie nun (immer noch bei gedrückter Shift-Taste) mit dem Selektieren fort, bis alle Piktogramme ausgewählt sind, die Sie bearbeiten wollen. Nun können Sie einen Menübefehl aufrufen, der alle selektierten Piktogramme nacheinander bearbeitet.

Sie können auf diese Weise auch eine Gruppe von Piktogrammen verschieben. Halten Sie dazu einfach die (linke) Maustaste fest, wenn Sie das letzte Piktogramm ausgewählt haben und bewegen Sie die Maus. Alle zuvor selektierten Piktogramme folgen dann gleichzeitig den Mausbewegungen.

3.5.5 Wie Sie ein völlig verdecktes Fenster nach vorne holen

Zum Abschluß dieses Kapitels noch ein Tip, der ganz hilfreich sein kann, wenn Sie die Möglichkeit, Schubladen zu bilden, intensiv nutzen. Wenn allzu viele Fenster auf Ihrer Workbench liegen und Sie ein bestimmtes nicht mehr finden können, müssen Sie vielleicht alle anderen Fenster schließen oder mit dem Back-Gadget nach hinten legen, um das gesuchte zu finden. Sie können das gesuchte Fenster aber auch sehr leicht zum Vorschein bringen, wenn Sie zufällig die Schublade sehen können, aus der das Fenster hervorgegangen ist. Öffnen Sie einfach die Schublade noch einmal mit einem Doppelklick auf das Piktogramm, und das entsprechende Fenster erscheint oben auf dem Stapel. Ein kleiner Trick, mit dem man sich manchmal viel Arbeit erspart. (Wie Sie sehen, hat die elektronische Werkbank in diesem Fall deutliche Vorteile gegenüber einer wirklichen. Bei einer echten Werkbank können Sie eine schon offene Schublade natürlich nicht noch einmal öffnen.)

3.6 Die Menübefehle der Workbench

Im folgenden finden Sie eine Liste aller Menübefehle, die Sie auf der Workbench aus den Menüs wählen können. Jeder Befehl wird zunächst auf englisch aufgelistet, dann eine ungefähre deutsche Übersetzung oder Umschreibung. Die Menü-Namen (fettgedruckt und unterstrichen) sind den einzelnen Befehlen (fettgedruckt) jeweils vorangestellt.

Workbench

Open	Öffnet ein Objekt; zum Beispiel eine Schublade
Close	Schließt ein Disketten- oder Schubladenfenster
Duplicate	Dupliziert ein beliebiges Objekt
Rename	Zum Umbenennen eines Objektes
Info	Zeigt das Info-Fenster eines Objektes
Discard	Löscht ein Objekt unwiderruflich

Disk

Empty Trash	Löscht unwiderruflich alle Objekte im Mülleimer
Initialize	Bereitet eine neue Diskette zur Datenaufnahme vor und löscht alte Disketten

Special

Clean Up	Ordnet die Objekte in einer Diskette oder Schublade
Last Error	Zeigt die letzte Fehlermeldung noch einmal
Redraw	Neuzeichnen der Workbench nach Programmfehlern
Snapshot	Speichert die aktuelle Position eines Piktogramms und die Größe und Position seines Fensters auf Diskette ab
Version	Zeigt die Versionsnummer der Workbench-Software

4 Zusammenhänge: Werkzeuge und Projekte

Wenn Sie mit Ihrem Amiga wirklich etwas vollbringen wollen, so benötigen Sie dazu ein Programm. Dies ist bei keinem Computer anders und sollte inzwischen bereits ein recht vertrautes Prinzip für Sie sein. Denn auch bisher (in den ersten drei Kapiteln dieses Buches) haben wir uns ja bereits mit einem Programm beschäftigt, der Workbench. Die Workbench aber ist ein sogenanntes »Systemprogramm«, das von den meisten Anwendern als selbstverständliche Einrichtung mehr oder weniger übersehen wird. Spricht man von einem Programm, so meint man damit meist ein sogenanntes Anwendungsprogramm, wie zum Beispiel eine Textverarbeitung oder ein Buchhaltungsprogramm. Erst diese Anwendungsprogramme machen einen Computer zu einem wirklichen Hilfsmittel; ohne sie bleibt er ein teurer Türstopper.

4.1 Werkzeuge ohne Projekte: Uhr und Rechner

Auf der Workbench des Amiga heißen Anwendungsprogramme »Werkzeuge«, und genau das sollten gute Programme auch sein. Kein Programm leistet wirklich »von selbst« etwas. Es kann Ihnen aber, wie ein gutes Werkzeug, dabei helfen, eine Aufgabe zu erledigen, beziehungsweise die Erledigung dieser Aufgabe erst zu ermöglichen. Sie werden nun zwei sehr einfache Werkzeuge kennenlernen.

4.1.1 Wie Sie ein Werkzeug starten

Im letzten Kapitel wurden die meisten Operationen mit Werkzeugen und Projekten bereits vorgestellt. Die eine Operation, die noch nicht erläutert wurde, war das Öffnen von Werkzeugen – alle anderen Objekte auf der Workbench kann man ja schließlich mit dem Befehl *Open* oder einem Doppelklick öffnen. Bereits bei diesen anderen Objekten galt immer die Grundregel, daß das Öffnen eines Objektes immer zeigt, »was sich darin verbirgt«. Um zu sehen, was sich in einem Programm verbirgt, muß man es natürlich starten – und genauso wirkt der Befehl *Open* auch, er startet das »geöffnete« Programm.

Um ein Werkzeug (Programm) zu starten, selektieren Sie es einfach mit einem Mausklick und wählen dann den Befehl *Open* aus dem Menü *Workbench*. Alternativ können Sie dasselbe auch durch einen Doppelklick mit der Maus auf das Piktogramm des Werkzeugs erreichen. Probieren Sie es aus!

Auf der Workbench-Diskette liegt direkt im Diskettenfenster ein Piktogramm, das wie eine Uhr aussieht und auch den vielsagenden Namen *Clock* (engl. für *Uhr*) trägt. Dies ist ein Werkzeug. Starten Sie es nun mit einem Doppelklick auf dieses Piktogramm. Das Diskettenlaufwerk beginnt zu arbeiten, liest das Programm von der Diskette in den RAM-Speicher des Amiga und startet es dann.

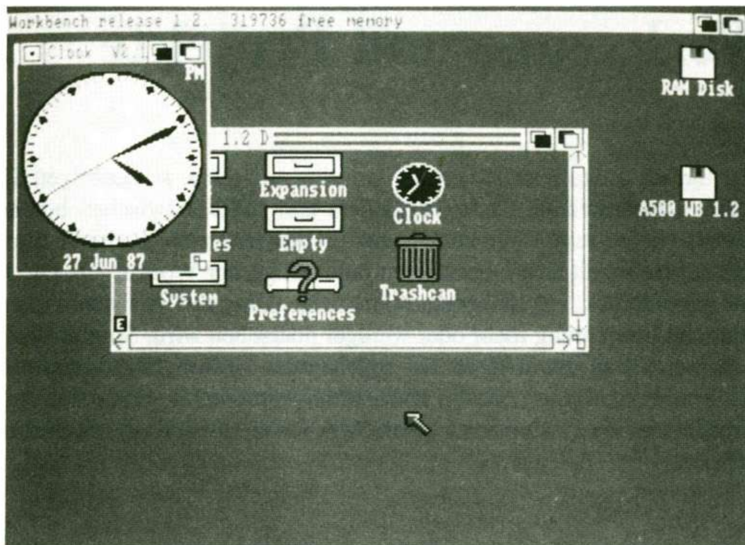


Bild 4.1: *Die Uhr des Amiga*

Nach wenigen Sekunden erscheint ein neues Fenster links oben auf der Werkbank. Es trägt den Titel *Clock V2.15* und zeigt in seinem Innern ein wirklichkeitsgetreues Bild einer (analogen) Uhr mit sich langsam bewegendem Sekundenzeiger. Am unteren Fensterrand wird zudem auch noch das Datum angezeigt. Uhrzeit und Datum stimmen vielleicht nicht, ansonsten geht die Uhr aber genau. Der Amiga 500 hat standardmäßig zwar eine ziemlich genau gehende Uhr eingebaut, aber keine Batterie, die dafür sorgt, daß diese Uhr auch dann noch weiterläuft, wenn Sie das Gerät abgeschaltet haben. Nur wenn Sie eine Amiga-501-Speichererweiterungskarte eingebaut haben, die auch eine batteriegepufferte Uhr enthält, stimmt die Uhrzeit, wenn Sie sie einmal richtig eingestellt haben, immer. Falls Sie diese Speichererweiterungskarte nicht besitzen, ist es nach jedem Neustart nötig (und auch ratsam) die Uhr nachzustellen. Sie können dies mittels des Werkzeugs *Preferences* machen, das weiter unten noch erläutert wird.

Sie können die Uhr einfach in der Ecke des Bildschirms liegen lassen und mit Ihrer Arbeit auf der Workbench fortfahren. Die Uhr läuft dabei weiter (wie es sich für eine ordentliche Uhr gehört), und zeigt Ihnen weiter die falsche Zeit exakt an (wie es sich für eine ordentliche Uhr nicht gehört).

4.1.2 Das Fenster eines Werkzeugs

Sobald Sie das Fenster der Uhr anklicken, zum Beispiel um es zu bewegen, teilen Sie dem Programm *Clock* mit, daß Sie damit arbeiten beziehungsweise direkt mit ihm kommunizieren wollen. Dies wird durch zwei Änderungen auch am Bildschirm verdeutlicht. Zum einen wird das Fenster aktiv: der Titelstreifen wird besser lesbar (er enthält die gepünktelte, schlecht lesbare Geisterschrift, solange das Fenster inaktiv ist). Zum anderen ändert sich die Titelleiste des Bildschirms. Statt *Workbench 1.2* und der Angabe des freien Speicherplatzes steht dort nun auch *Clock V2.15*.

Sobald Sie nämlich ein Werkzeug öffnen, wird es wichtig, ob ein Fenster aktiv ist oder nicht. Solange das Fenster eines Programms nämlich inaktiv ist, kann das Programm zwar tätig sein und auch Ausgaben in seinem Fenster zeigen. Aber erst wenn das Fenster aktiv wird, können Sie Einfluß auf das Programm nehmen. Da Sie sich nicht zweiteilen können, können Sie ja immer nur mit einem Programm zur selben Zeit kommunizieren. Nur dieses eine Programm empfängt auch Ihre Aktionen (zum Beispiel Mausklicks und Tastendrücke). Indem Sie auf das Fenster eines bestimmten Programms klicken (es also aktiv machen), schenken Sie diesem Programm Ihre Aufmerksamkeit. Das Programm, das vorher aktiv war, geht dann in eine Art Wartezustand über, in dem es ohne Ihre Eingaben (zum Beispiel Mausklicks und Tastendrücke) auskommen muß. Das neue Programm empfängt nun Ihre Eingaben und reagiert darauf. Wie es darauf reagiert, entscheidet es allein. Deshalb kann auch ein Mausklick oder eine Bewegung der Maus in einem Fenster etwas völlig anderes bewirken als in einem anderen Fenster.

Die Systemsoftware des Amiga speichert zu jedem Fenster, das am Bildschirm sichtbar ist, ab, zu welchem Programm es gehört. So kann sie jederzeit entscheiden, welches Programm die »Nachrichten«, die vom Anwender kommen, empfangen und verarbeiten soll.

Experimentieren Sie doch ein wenig mit dem Uhrfenster! Sie werden sehen, daß es sich ganz ähnlich verhält wie die Fenster der Workbench, die wir in den vorangegangenen Kapiteln schon kennengelernt haben. Das Uhrfenster enthält zum Beispiel ein Größen-Gadget, Back- und Front-Gadgets und ein Schließ-Gadget, mit dem Sie das Fenster schließen und damit das Programm (Werkzeug) *Clock* wieder beenden können. Versuchen Sie vor allem einmal das Größen-Gadget. Die Uhr paßt sich jeder neuen Fenstergröße an. Wenn Sie wollen, können Sie eine Uhr in Bildschirmgröße haben. Wird das Fenster rechteckig, so wird das Ziffernblatt entsprechend oval.

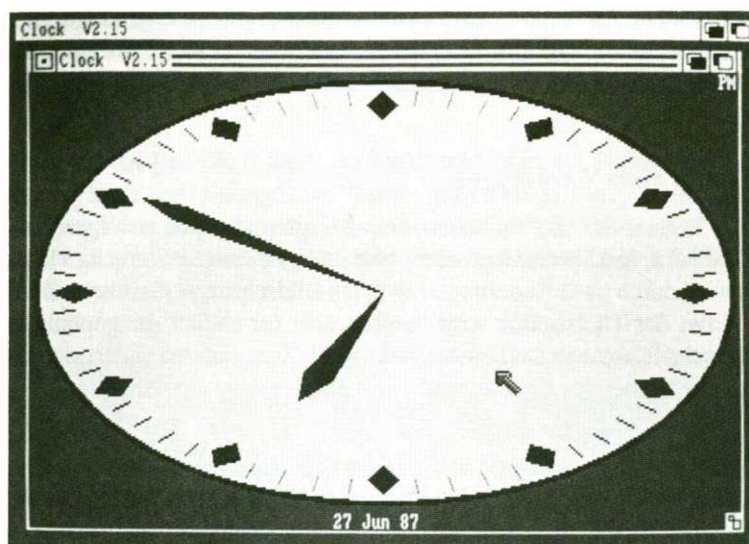


Bild 4.2: Eine riesige Uhr

4.1.3 Die Menüleiste eines Werkzeugs

Wie oben bereits erwähnt, wechselt jedesmal, wenn man auf das Fenster eines anderen Programms klickt, der Inhalt der Titelleiste des Bildschirms. Diese Titelleiste enthält aber nicht nur einen kurzen Namen wie *Workbench* oder *Clock*, sondern »darunter« auch die Menüs, die Ihnen aus den letzten Kapiteln ja schon bekannt sind. Auch diese Menüs werden jedesmal komplett ausgetauscht, wenn Sie ein Fenster eines anderen Programms anklicken. Drücken Sie die Menütaste, während das Fenster der Uhr aktiv ist, und Sie werden fünf neue Menütitel sehen: *Type*, *Mode*, *Seconds Date* und *Alarm*.

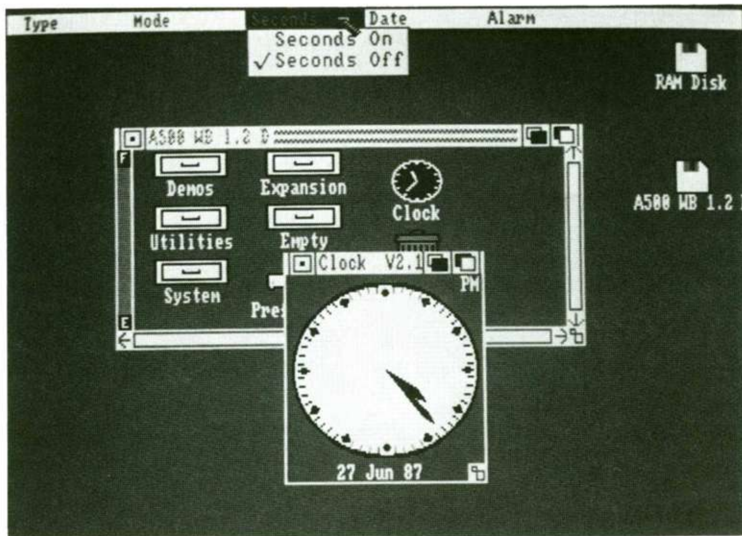


Bild 4.3: *Die Menüleiste eines Werkzeugs*

Im Menü *Type* stehen Ihnen drei Befehle zur Auswahl: *Digital1*, *Digital2* und *Analog*. Der Befehl *Analog* ist zusätzlich noch mit einem »Häkchen« versehen (»abgehakt«), das Ihnen sagt, daß die Uhr im Moment auf Analog-Anzeige geschaltet ist (was Sie natürlich auch ohne den Haken sehen würden). Dadurch lernen Sie auch gleich einen weiteren Aspekt von Menüs kennen: Gibt es nämlich in einem Menü einen Befehl, der quasi ein Schalter ist, so kann die Stellung des Schalters durch ein Häkchen angezeigt werden. Ist der Schalter an (Analog-Anzeige sichtbar), wird der Befehl mit einem Haken versehen (abgehakt), ansonsten nicht. Wenn Sie den Befehl *Digital1* im *Type*-Menü wählen, wechselt die Anzeige der Uhrzeit von Zeigern nach Ziffern. Danach ist *Digital1* mit einem Häkchen versehen und *Analog* nicht mehr. Die beiden Digitalanzeigen haben allerdings Fenster von fester Größe, so daß es beim Wechsel von einer Anzeigeart zur anderen meist auch zu einem Sprung in der Fenstergröße kommt, der etwas überraschend sein kann.

Der letzte Befehl, *Digital2*, zeigt eine andere, sehr kompakte Form der Digitalanzeige. Während die Digital-Uhr von *Digital1* noch zweizeilig ist und gleichzeitig Uhrzeit und Datum anzeigt, besteht sie bei *Digital2* nur noch aus der Titelleiste des Fensters, in der abwechselnd Datum und Uhrzeit zu sehen sind.

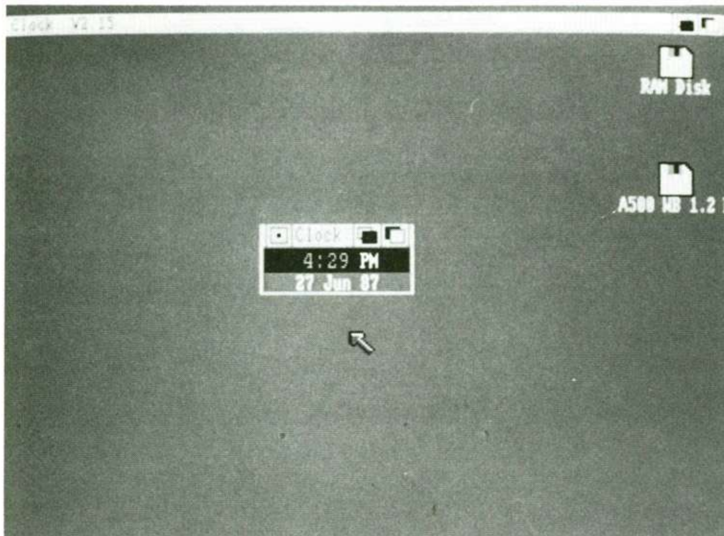


Bild 4.4: *Die Digitalanzeige der Uhr des Amiga*

Im Menü *Mode* können Sie zwischen 24-Stunden-Anzeige und 12-Stunden-Anzeige für die Uhr wählen, was natürlich nur für die Digital-Uhr von Bedeutung ist. Die Bedienung ist genauso wie bei *Type*. Der jeweils zuletzt gewählte – und damit gültige – Befehl wird mit einem Häkchen versehen. Genauso können Sie im Menü *Seconds* mit *Seconds Off* den Sekundenzeiger beziehungsweise die Sekundenanzeige der Digitaluhr abschalten und mit *Seconds On* wieder anschalten. Und im Menü *Date* schließlich können Sie in entsprechender Weise die Datumsanzeige an- und abschalten (mit *Date On* beziehungsweise *Date Off*)

Im letzten Menü *Alarm* können Sie die Uhr des Amiga schließlich auch noch als Wecker einsetzen. Unten in diesem Menü gibt es die beiden Punkte *Alarm On* und *Alarm Off*, von denen wieder der jeweils gültige mit einem Häkchen versehen ist. Wie sich das für einen richtigen Wecker gehört, klingelt auch der Wecker des Amiga auf keinen Fall, solange *Alarm Off* gültig (abgehakt) ist. Nur wenn *Alarm On* abgehakt ist, kann der Wecker klingeln. Mit dem ersten Befehl *Set* in diesem Menü können Sie die Alarm- beziehungsweise Weckzeit einstellen. Rufen Sie diesen Befehl auf und es erscheint ein Fenster mitten auf dem Bildschirm.

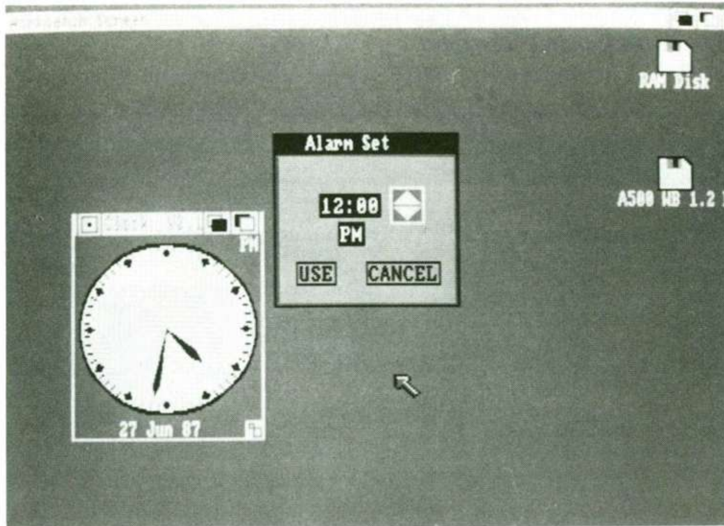


Bild 4.5: Einstellen der Alarmzeit

In diesem Fenster sehen Sie unter anderem die Uhrzeit 12:00 Uhr in digitaler Form. Klicken Sie auf die angezeigten Stunden (also die Zahl 12) und Sie können mit den danebenliegenden Pfeilsymbolen die Stunden vor- und zurücklaufen lassen. Wenn Sie auf den nach oben weisenden Pfeil klicken, werden die Zahlen größer. Wenn Sie auf den nach unten weisenden Pfeil klicken, werden die Zahlen kleiner. Um die Minuten der Alarmzeit einzustellen, müssen Sie zunächst auf die Minutenanzeige (also die Zahl 00) klicken und können dann die Minutenzahlen genau wie die Stunden mit den beiden Pfeilen vergrößern und verkleinern.

Um das Fenster wieder verschwinden zu lassen, müssen Sie auf einen der Knöpfe USE (engl. für *Verwenden*) oder CANCEL (engl. für *Abbrechen*) klicken. Nur wenn Sie auf USE klicken, wird die eingestellte Weckzeit gültig, bei CANCEL ignoriert der Amiga den ganzen Vorgang. Sie werden aber natürlich nur dann zur eingestellten Zeit mit einem Piepston »geweckt«, wenn *Alarm On* im Menü *Alarm* eingestellt ist.

4.1.4 Der Rechner

Die Amiga-Uhr ist nicht gerade ein überwältigendes, aber doch ein ganz nettes kleines Programm, für das Sie gelegentlich sicher Verwendung finden werden. Sie werden nun ein ähnlich einfaches Werkzeug kennenlernen, das Ihnen manchmal vielleicht eine echte Hilfe sein kann: den Taschenrechner.

Öffnen Sie dazu (natürlich wieder mit einem Doppelklick) die Schublade Utilities auf der Workbench-Diskette. Darin befinden sich zwei Piktogramme: *Notepad* und *Calculator* (engl.

für *Rechner*). Machen Sie einen Doppelklick auf dem Piktogramm *Calculator* und der Taschenrechner erscheint am Bildschirm.

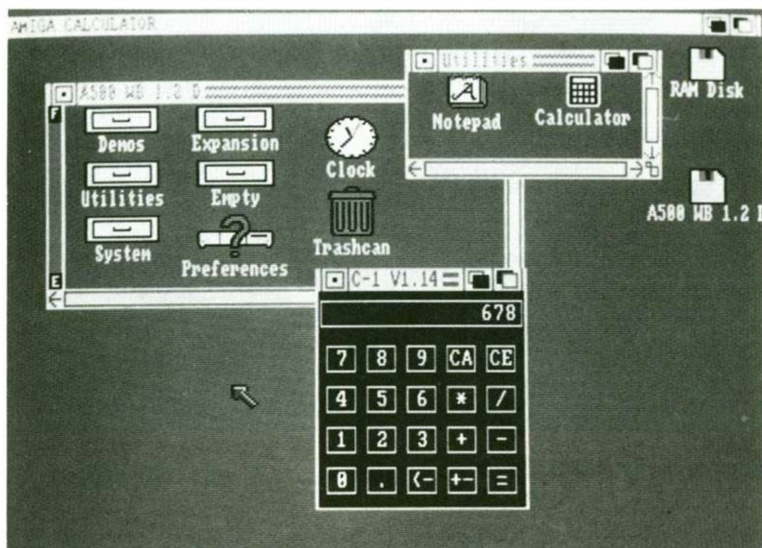


Bild 4.6: Der Taschenrechner des Amiga

Es handelt sich dabei um einen sehr simplen Taschenrechner, der nur die vier Grundrechenarten beherrscht und nicht einmal über einen Speicher verfügt. Dafür können Sie ihn sowohl mit der Maus als auch über die Tastatur bedienen. Die »Tasten« des Taschenrechners sind Gadgets, die Sie mit einem Mausklick betätigen können. Wollen Sie *2 * 3* berechnen, können Sie zum Beispiel die Taste [2], dann [*], dann [3] und schließlich die [=]-Taste anklicken. Alternativ dazu können Sie aber auch *2 * 3* auf der Tastatur eingeben und dann die [=]-Taste, die [Enter]-Taste oder die [Return]-Taste auf der Tastatur drücken.

Abgesehen von der Möglichkeit, sowohl die Tastatur als auch die Maus zur Bedienung zu benutzen, verhält sich der Taschenrechner wie ein normaler Taschenrechner auf Ihrem Schreibtisch. Auch die restlichen »Tasten« sollten Ihnen bereits vertraut sein. [CA] löscht sämtliche Eingaben. [CE] löscht nur die zuletzt eingegebene Zahl, aber nicht ein vorher bereits errechnetes Zwischenergebnis. Und mit [<-] löschen Sie die zuletzt eingegebene Ziffer. (Diese Taste wirkt also wie die [Backspace]-Taste bei der Texteingabe.) Fast alle »Tasten« des Taschenrechners haben ein gleichbedeutendes Pendant auf der Tastatur. [<-] entspricht zum Beispiel der [Backspace]-Taste. Für [CA] und [CE] müssen Sie zwei Tasten auf der Tastatur unmittelbar hintereinander drücken: erst [C] und dann [A] beziehungsweise [E].

Wenn Sie den Rechner nicht mehr benötigen, brauchen Sie nur sein Fenster zu schließen. Klicken Sie dazu in das Schließ-Gadget in der oberen linken Ecke des Fensters.

4.2 Ein Werkzeug für Projekte: Notepad

Die beiden Werkzeuge (Programme), die Sie bisher kennengelernt haben, erzeugen oder bearbeiten keine Dokumente beziehungsweise Dateien. (Welches Projekt wollte man auch mit dem »Werkzeug Uhr« bearbeiten?) In diesem Abschnitt werden Sie nun ein Werkzeug kennenlernen, das Dateien erzeugen und bearbeiten kann. Es handelt sich um das Werkzeug *Notepad* (engl. für *Notizblock*), ein kleines Textverarbeitungsprogramm. Sie verfügen mit *Notepad* zwar nicht über eine besonders leistungsfähige Textverarbeitung, können damit aber recht komfortabel kurze Texte (Textdateien) bearbeiten. Das ist für das Anlegen kleiner Notizen und zum Herumspielen mit den Möglichkeiten der Textverarbeitung auf dem Amiga aber völlig ausreichend.

Diese Texte sind die »Projekte« von *Notepad.* Alle Dateien, die von einem Programm (Werkzeug) erzeugt werden, heißen auf der Workbench »Projekte«. In der Ihnen inzwischen ja schon bekannten Werkbank-Analogie kann man sich das so vorstellen, daß mit einem bestimmten Werkzeug an einem bestimmten (oder mehreren) Projekt(en) gearbeitet wird. Diese Verbindung zwischen Projekten und Werkzeugen wird meist auch durch die Piktogramme der entsprechenden Workbench-Objekte verdeutlicht. Das Piktogramm eines Projekts, das zu einem bestimmten Werkzeug gehört, sieht diesem Werkzeug meist ähnlich, wie Sie unten auch am Beispiel sehen werden.

4.2.1 Wie Sie den Notizblock öffnen

Um an den Notizblock »heranzukommen«, müssen Sie zunächst wieder die Schublade *Utilities* auf der Workbench-Diskette öffnen. In dem Fenster, das sich daraufhin öffnet, sehen Sie dann das Piktogramm *Notepad*, hinter dem sich der Notizblock »verbirgt«. Sie können den Notizblock nun genauso einfach öffnen wie die Uhr. Dazu reicht ein Doppelklick mit der Maus auf das Piktogramm. Das Diskettenlaufwerk rotiert nun aber schon merklich länger als beim Öffnen der Uhr, bis endlich ein neues Fenster mit dem Namen *Notepad V2.0* erscheint. Im Gegensatz zur Uhr wird dieses Fenster sofort aktiv und auch die Titelleiste des Bildschirms zeigt gleich durch den neuen Inhalt *Notepad* an, daß dieses Programm nun Ihre Eingaben erwartet. Sie werden nun zunächst erfahren, wie Sie einen Text in diesen Notizblock eintragen und ändern können.

4.2.2 Wie Sie Text in den Notizblock eintragen

Nach dem Öffnen von *Notepad* wird Ihnen zunächst ein leeres weißes Blatt gezeigt, dessen Größe Sie mit Hilfe des Größen-Gadgets jederzeit ändern können. Oben links auf diesem weißen Blatt sehen Sie einen dünnen schwarzen Strich. Tippen Sie nun auf der Tastatur ein paar Buchstaben, so erscheinen diese immer links neben der Stelle, an der der Strich gerade steht und der Strich wandert dabei nach rechts. Bei jedem Buchstaben, der neu eingetippt wird, wandert der Strich nach rechts weiter. Es handelt sich bei ihm natürlich um eine sogenannte »Schreibmarke« (oft auch »Cursor« genannt), die Sie ja schon kennengelernt haben (beim Befehl *Rename*). Diese Schreibmarke steht – im Gegensatz zu der beim *Rename*-Befehl oder in einem Infofenster – immer zwischen zwei Buchstaben. Neu eingetippter Text erscheint immer zwischen diesen beiden Buchstaben und schiebt den Text rechts davon zur Seite.

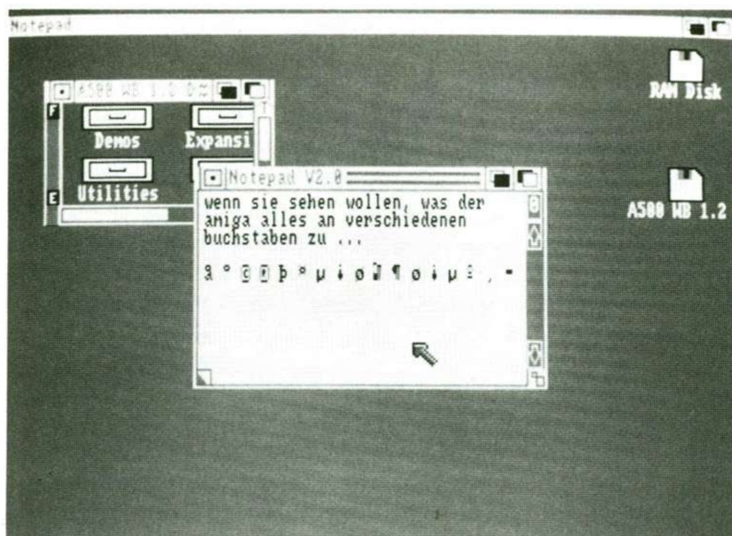


***Bild 4.7:** Eingabe eines Textes auf dem Notizblock des Amiga*

Wenn Sie sehen wollen, was der Amiga so alles an verschiedenen Buchstaben zu bieten hat, so halten Sie einmal eine der beiden [Alt]-Tasten fest und tippen dann auf eine der »normalen« Buchstaben-Tasten – genauso wie Sie die [Shift]-Taste festhalten müssen, um Großbuchstaben zu bekommen. Sie können mit der [Alt]-Taste eine Vielzahl von speziellen Buchstaben und Sonderzeichen, wie ñ, œ, ç, µ und ∂, erzeugen, so daß Sie auch bei Briefen an Ihre französischen, spanischen oder norwegischen Freunde nie in die Verlegenheit kommen werden, einen Buchstaben der Landessprache nicht verwenden zu können.

Die Schreibmarke wandert aber nicht nur »von selbst« weiter, während Sie Text eintippen oder löschen. Wenn Sie mit der Maus an die Stelle zwischen zwei Buchstaben klicken, so springt die Schreibmarke zu dieser Stelle. Text, den Sie danach eintippen, erscheint dann dort. Während Sie gerade Ihre Hände auf der Tastatur haben, können Sie die Schreibmarke aber vielleicht noch bequemer mit den Pfeil-(Cursor-)Tasten bewegen. Jede dieser vier Tasten bewegt die Schreibmarke in die angegebene Richtung, »solange es geht«. Das heißt, Sie können nur so lange mit der [Dn]-Taste nach unten gehen, bis das Ende des Textes erreicht ist, die [Up]-Taste »stößt« genauso in der ersten Zeile an.

4.2.3 Wie Sie kleine Fehler korrigieren

Machen Sie Fehler bei der Texteingabe, so können Sie diese mit der [Backspace]-Taste (oben rechts auf der Tastatur) wieder löschen. Die [Backspace]-Taste löscht immer den ersten Buchstaben links von der Schreibmarke. Die restlichen Buchstaben rechts davon rücken dabei nach links auf. Wenn Sie einen falschen Buchstaben eingeben und dies sofort bemerken, so brauchen Sie nur einmal auf [Backspace] zu drücken und der Fehler ist beseitigt. Wenn Sie den

Fehler erst bemerken, wenn Sie schon weiteren Text eingegeben haben, müssen Sie die Schreibmarke bewegen. Klicken Sie mit der Maus hinter den fehlerhaften Buchstaben oder betätigen Sie die Cursortasten, bis die Schreibmarke hinter dem Fehler steht, und drücken dann [Backspace]. Durch mehrmalige Betätigung dieser Taste können Sie natürlich auch längere Textpassagen löschen.

Ähnlich wie die [Backspace]-Taste funktioniert die Taste [Del] (rechts neben der [Backspace]-Taste). Sie löscht den Buchstaben rechts neben der Schreibmarke sofern dort einer steht.

4.2.4 Wie Sie im Text blättern

Wird das Fenster voll, so können Sie auf ein neues Blatt überwechseln. Hierzu dienen die beiden Gadgets oben rechts und unten links im Fenster. Das Gadget unten links, das wie ein »Eselsohr« aussieht, blättert zur nächsten Seite, bis die letzte (zehnte) Seite erreicht ist. Das Gadget oben rechts, in dem eine Zahl steht, zeigt bei Betätigung (anklicken) die jeweils vorangehende Seite. In diesem Gadget wird auch immer die Nummer der aktuellen Seite angezeigt. Sie können sich also jederzeit vergewissern, auf welcher der zehn Seiten des Notizblocks Sie sich befinden.

Es ist auch möglich, mehr Text auf ein Blatt zu tippen, als ins Fenster paßt. Der Fensterinhalt wird dann nach oben verschoben und geht dabei auch nicht verloren. Sie können ihn jederzeit mit Hilfe des Rollbalkens am rechten Fensterrand wieder zum Vorschein bringen. Dieser funktioniert genauso wie die Rollbalken eines Schubladenfensters. Klicken Sie einmal auf den Pfeil am unteren Ende des Rollbalkens und der Fensterinhalt wird nach oben verschoben, wodurch eventuell am unteren Rand bislang verborgene Zeilen auftauchen. Klicken Sie einmal auf den Pfeil am unteren Ende des Rollbalkens und der Fensterinhalt wird nach unten verschoben, wodurch eventuell am oberen Rand verschwundene Zeilen wieder auftauchen. Wenn Ihnen dieses Hin- und Herschieben zu lästig ist, können Sie aber natürlich auch das Fenster des Notizblocks größer machen. Sie können dann einen größeren Teil Ihres Textes bearbeiten, ohne zur Maus greifen zu müssen.

Sie sollten nun schon wissen, wie Sie Texte eingeben, sich in einem größeren mehrseitigen Text zurechtfinden und kleine Fehler darin korrigieren können. Besonders die Hand- (beziehungsweise Maus-)Griffe für das Eingeben und Korrigieren von Texten sollten Sie sich unbedingt einprägen. Sie werden diese Fertigkeiten in fast allen anderen Programmen auch benötigen!

4.2.5 Wie Sie Textpassagen selektieren

Das Korrigieren größerer Fehler mit Hilfe der Tasten [Backspace] und [Del] ist allerdings recht umständlich. Hierfür gibt es die Befehle im *Edit*-Menü, mit denen Sie längere Textpassagen löschen, kopieren und innerhalb des Textes verschieben können. Um diese Befehle benutzen zu können, müssen Sie aber zunächst wissen, wie Sie eine Textpassage im Notizblock selektieren können. Das hat eine ähnliche Aufgabe wie die Selektion eines Piktogramms auf der Workbench: Sie teilen dem Programm dadurch mit, auf welchen Teil des Textes der nächste Befehl angewendet werden soll.

Zum Selektieren müssen Sie die Schreibmarke zunächst vor den ersten Buchstaben der zu selektierenden Textpassage setzen und dann den Befehl *Mark Place* aus dem Menü *Edit* wählen. Setzen Sie dann die Schreibmarke (mit der Maus oder den Cursortasten) an das Ende der Textpassage, die Sie selektieren wollen, und wählen Sie erneut *Mark Place* aus dem Menü *Edit*. Der so markierte Bereich wird nun »hervorgehoben«, das heißt, er erhält eine andere Hintergrundfarbe. Normalerweise erscheint hervorgehobener Text im Notizblock schwarz auf orange statt schwarz auf weiß.

Probieren Sie es aus: Tippen Sie die Worte »Das ist mein erster Satz« ein und setzen Sie die Schreibmarke dann vor das Wort »ist«. Wählen Sie den Befehl *Mark Place* aus dem *Edit*-Menü und setzen Sie die Schreibmarke hinter das Wort »mein«. Wenn Sie nun noch einmal *Mark Place* aufrufen, werden die Worte «ist« und »mein« hervorgehoben.

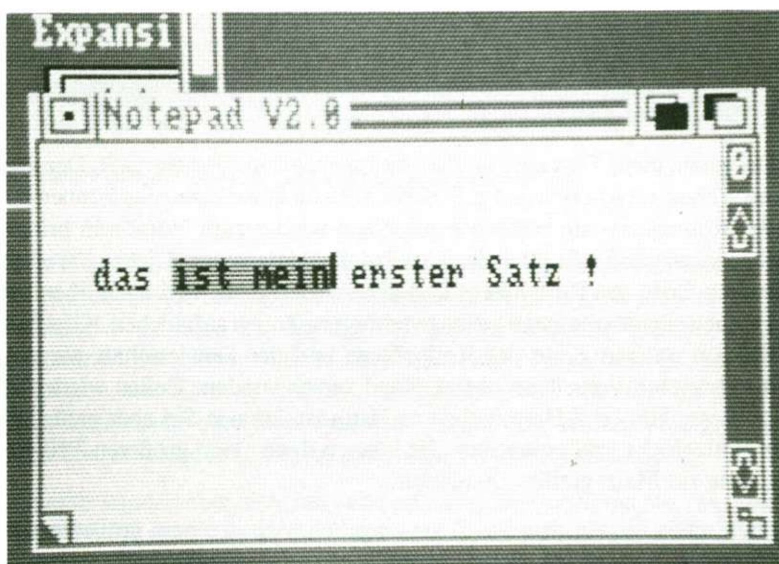


***Bild 4.8:** Hervorheben einer selektierten Textpassage im Notizblock*

Mark Place hat allerdings noch zwei Abkürzungen – es ist schließlich recht umständlich, zweimal die Schreibmarke zu versetzen und zwei Menübefehle auswählen zu müssen, um eine Textpassage zu markieren. Der Menübefehl *Mark Place* – und auch mancher andere Befehl – hat deshalb ein »Tastaturäquivalent«. Ein Tastaturäquivalent ist eine Möglichkeit, durch das Drücken zweier bestimmter Tasten denselben Effekt zu erzielen wie durch die Auswahl eines Menübefehls. Wenn ein Menübefehl ein Tastaturäquivalent besitzt, so wird dies auch im Menü sichtbar gemacht. Klappen Sie dazu noch einmal das *Edit*-Menü herunter und schauen Sie es sich näher an. Rechts neben diesen Befehlen ist ein *A* auf dunklem Hintergrund und ein zweiter Buchstabe zu sehen. Das ist die Beschreibung eines Tastaturäquivalents; jeder Befehl im *Edit*-Menü besitzt eine solche Abkürzung. Die Bedeutung dieser zwei Buchstaben ist ganz einfach

zu verstehen: Neben dem Befehl Mark Place steht zum Beispiel ***A*M**. Das bedeutet, daß Sie die Amiga-Taste (*A*) rechts neben der Leertaste festhalten und dann die [M]-Taste drücken müssen, um diesen Befehl aufzurufen.

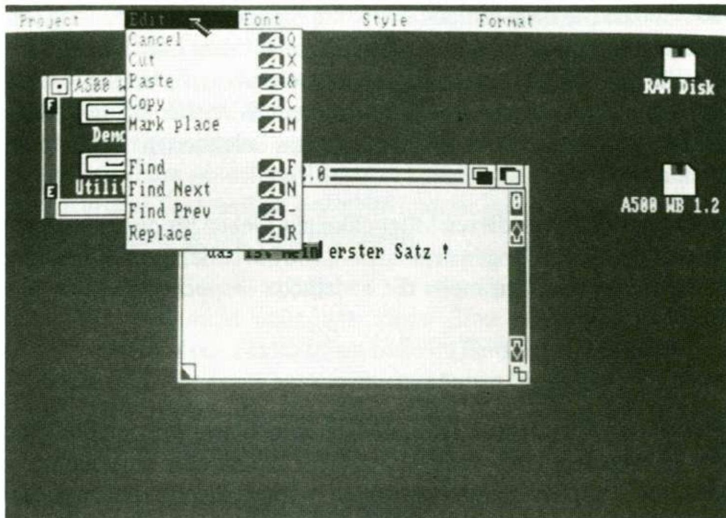


Bild 4.9: *Tastatur-Äquivalente in einem Menü*

Probieren Sie es aus: Verwenden Sie dazu wieder den Text »Das ist mein erster Satz«. Wählen Sie zunächst den Befehl *Cancel* aus dem *Edit*-Menü, wenn Sie bereits eine Textpassage hervorgehoben haben. Die Hervorhebung des selektierten Textes verschwindet dann. Setzen Sie dann die Schreibmarke vor das Wort »mein«. Drücken Sie nun auf die Amiga-Taste rechts neben der Leertaste und halten Sie sie fest. Betätigen Sie dann die Taste [M] und lassen Sie die beiden Tasten wieder los. Setzen Sie die Schreibmarke nun hinter das Wort »erster« und betätigen Sie noch einmal, wie eben beschrieben, die Tastenkombination [*A*]-[M]. Nun werden die beiden Worte »mein erster« hervorgehoben.

Wie Sie gerade gesehen haben, wirkt die Tastenkombination [*A*]-[M] also genauso wie die Auswahl von *Mark Place* aus dem *Edit*-Menü mit der Maus. Die dritte Möglichkeit, *Mark Place* zu »simulieren«, ist ein Doppelklick mit der Maus an eine bestimmte Stelle im Text.

Probieren Sie es aus: Wählen Sie zunächst wieder *Cancel* aus dem *Edit*-Menü, falls die Worte »mein erster« noch hervorgehoben sind. Bewegen Sie dann den Mauszeiger, so daß seine Spitze genau über oder etwas vor dem »m« des Wortes »mein« steht und drücken Sie schnell zweimal hintereinander auf die Selektionstaste (die **linke** Maustaste). Bewegen Sie dann die Mausspitze kurz hinter den Buchstaben »r« am Ende des Wortes »erster« und klicken Sie

wieder zweimal auf die Selektionstaste. Die Worte »mein erster« werden nun wieder hervorgehoben.

Sie können den Bereich des selektierten Textes, die »Selektion«, aber mit *Mark Place* nicht nur festlegen, sondern auch erweitern oder verkleinern, nachdem Sie schon eine Textpassage selektiert haben. Setzen Sie dazu einfach die Schreibmarke vor den Anfang oder hinter das Ende der Selektion und wählen Sie wieder *Mark Place* aus dem *Edit*-Menü (oder machen Sie dort einen Doppelklick). Die Stelle, an der sich die Schreibmarke befand, wird dadurch zum neuen Anfang beziehungsweise Ende des selektierten Bereichs. Genauso können Sie mit einem Doppelklick (oder *Mark Place*) innerhalb des selektierten Bereichs die Selektion verkleinern.

Üben Sie nun ruhig ein wenig mit diesen Möglichkeiten. Geben Sie dazu ein paar längere Sätze ein. Sie werden dann bald »den Bogen raushaben«, wie Sie beliebige Textbereiche selektieren können. Am bequemsten ist dazu meist die »Methode Doppelklick«, wie Sie sicher bald merken werden.

4.2.6 Wie Sie Texte umstellen

Jetzt, da Sie wissen, wie Sie längere Textpassagen selektieren, können Sie Wörter, Absätze oder auch lange Textpassagen mit wenigen Schritten ändern oder umstellen. Wenn Sie zum Beispiel eine längere Textpassage löschen wollen, müssen Sie sie zunächst selektieren, also hervorheben, wie gerade beschrieben, und dann den Befehl *Cut* (engl. *to cut = ausschneiden*) aus dem Menü *Edit* wählen. Die zuvor hervorgehobene Textpassage verschwindet daraufhin – allerdings nicht auf Nimmerwiedersehen. »Ausgeschnittener« Text landet in der »Zwischenablage« (engl. *clipboard*). Dies ist ein kurzlebiger Zwischenspeicher, aus dem Sie versehentlich gelöschten Text wieder hervorholen können, den Sie aber auch dazu verwenden können, um Texte umzustellen oder zu duplizieren.

Sobald Sie nämlich den Befehl *Paste* (engl. *to paste = einkleben*; im übertragenen Sinn auch *einfügen*) aus dem Menü *Edit* wählen, wird der Text, der sich in der Zwischenablage befindet, an der Stelle in den Text eingefügt, an dem sich gerade die Schreibmarke befindet. Wenn Sie also eine Textpassage zu einer anderen Stelle im Text bewegen wollen, so müssen Sie diese Textpassage mit *Cut* ausschneiden, die Schreibmarke an die gewünschte andere Stelle setzen und *Paste* aufrufen.

Paste leert aber die Zwischenablage nicht! Sie können also, indem Sie *Paste* zweimal hintereinander aufrufen, zwei identische Kopien der Textpassage aus der Zwischenablage in den Text einfügen. Zum Kopieren längerer Textpassagen wird normalerweise jedoch der Befehl *Copy* verwendet. Auch hierzu müssen Sie wieder den Bereich des Textes, den Sie kopieren wollen, hervorheben und dann *Copy* (engl. *to copy = kopieren*) aus dem Menü *Edit* wählen. Dadurch gelangt der markierte Text in die Zwischenablage (und ersetzt deren alten Inhalt), wird aber nicht aus der Seite gelöscht. Sie können diese Textpassage nun beliebig oft auf dieser oder auf den anderen Seiten des Notizblocks einfügen, indem Sie die Schreibmarke an die gewünschte Stelle setzen und *Paste* aufrufen.

4.2.7 Wie Sie Texte suchen und ersetzen

Neben diesen Möglichkeiten, Texte »von Hand« zu ändern, bietet Notepad aber – wie jedes Textverarbeitungsprogramm, das etwas auf sich hält, eine Suche-und-Ersetze-Funktion. Sie können damit nach einem Wort oder einem Wortstück suchen und es gegebenenfalls gleich durch eine andere Buchstabenfolge ersetzen lassen. Hierzu dienen die Befehle *Find* (engl. *to find = finden*), *Find Next* (*= finde Nächstes*), *Find Prev* (*= finde Vorangehendes*) und *Replace* (engl. *to replace = ersetzen*) im *Edit*-Menü. Mit dem Befehl *Find* legen Sie zunächst fest, wonach Sie suchen und wodurch Sie es gegebenenfalls ersetzen wollen, *Find Next* sucht das nächste Vorkommen des gesuchten Textes in Richtung auf das Textende und *Find Prev* sucht das nächste Vorkommen des gesuchten Textes in Richtung auf den Textanfang. Mit *Replace* schließlich kann man eine Textpassage, nachdem man sie gefunden hat, durch eine andere, die man zuvor mit *Find* festgelegt hat, ersetzen lassen.

Probieren Sie es aus: Geben Sie einen längeren Text ein, in dem mehrfach das Wort »Suche« vorkommt (der Text muß nicht unbedingt einen Sinn ergeben). Setzen Sie dann die Schreibmarke an den Anfang des Textes (oben links im Fenster) und wählen Sie *Find* aus dem *Edit*-Menü. Statt *Find* mit der Maus auszuwählen, können Sie natürlich auch sein Tastaturäquivalent [*A*]-[F] verwenden – wie das geht, wissen Sie ja jetzt. Daraufhin erscheint ein Requester am Bildschirm, in dem Sie nun die Angaben machen müssen, die *Notepad* für die Suche braucht.

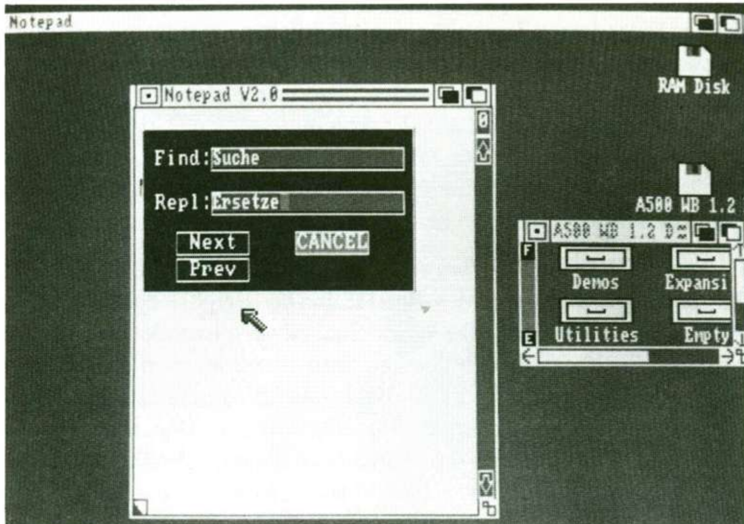


Bild 4.10: Der Find-Requester

Der Requester enthält zwei weiß umrandete blaue Rahmen in der oberen, und drei Knöpfe, die man mit der Maus betätigen kann, in der unteren Hälfte. Diese Rahmen sind »Textgadgets«. Sie können einen Text darin eingeben oder – ganz ähnlich wie im *Notepad* – ändern. Ähnliche Textgadgets haben Sie bereits im Zusammenhang mit dem *Rename*-Befehl der Workbench oder im Info-Fenster eines Piktogramms kennengelernt. In diesem Fall dienen die beiden Textgadgets dazu, einen Suchbegriff einzugeben und, sofern Sie das wünschen, einen Text, der diesen Suchbegriff ersetzen soll.

Klicken Sie dazu zunächst einmal mit der Maus in das Feld neben der Beschriftung *Find* und geben Sie das Wort »Suche« ein. Klicken Sie dann in das Textgadget neben der Beschriftung *Repl* (Abkürzung von *Replace*) und geben Sie »Ersetze« ein. Sie haben nun drei Möglichkeiten, den Requester wieder zu schließen. Wenn Sie den Knopf CANCEL anklicken, verschwindet der Requester einfach und nichts weiter geschieht. Klicken Sie jedoch *Next* an, so verschwindet der Requester und Notepad sucht nach dem ersten Vorkommen des neben *Find* eingegebenen Begriffs hinter der augenblicklichen Position der Schreibmarke. Wird ein solcher Begriff gefunden, befindet sich die Schreibmarke danach direkt vor dem ersten Buchstaben dieses Begriffs. Wenn Sie den Requester statt mit *Next* durch Anklicken von *Prev* (Abkürzung von *Previous = vorangehend*) schließen, sucht *Notepad* ebenfalls nach dem Suchbegriff, jedoch ausgehend von der augenblicklichen Position der Schreibmarke rückwärts auf den Anfang des Textes zu.

Klicken Sie nun auf *Next*. Die Schreibmarke müßte danach direkt vor dem ersten Vorkommen von »Suche« im Fenster des Notizblocks stehen. Wenn Sie nun den Befehl *Find Next* aus dem *Edit*-Menü wählen (oder dessen Tastaturäquivalent drücken), springt die Schreibmarke an den Anfang des nächsten Vorkommens von »Suche«. Wählen Sie dann *Find Prev* (ious), springt die Schreibmarke zurück zum vorangehenden (also ersten) Vorkommen von »Suche«. So können Sie nacheinander sehr rasch alle Vorkommnisse eines bestimmten kurzen Textes auffinden und zwischen diesen Stellen hin- und herspringen. Steht die Schreibmarke nach Aufruf eines dieser Suchbefehle vor einem Vorkommen des Suchbegriffs im Text, können Sie diesen mit einem Befehl gegen eine andere Buchstabenfolge austauschen.

Probieren Sie es aus: Wählen Sie *Find Next* oder *Find Prev* aus dem *Edit*-Menü, bis die Schreibmarke vor einem »Suche« steht. Rufen Sie nun den Befehl Replace aus dem *Edit*-Menü auf (oder drücken Sie [*A*]-[R]) und das Wort »Suche« verschwindet und statt dessen taucht »Ersetze« auf. Der Suchbegriff, den Sie im oben beschriebenen Requester neben *Find* eingetragen haben, wird also durch die Buchstabenfolge ersetzt, die Sie neben *Repl* eingetragen haben, sobald Sie den Befehl *Replace* aufrufen. Beachten Sie aber bitte, daß *Replace* nur dann funktioniert, wenn Sie den Suchbegriff zuvor gesucht (und gefunden) haben. Wenn Sie die Schreibmarke mit der Maus vor »Suche« plazieren und dann *Replace* aufrufen, geschieht nichts.

4.2.8 Wie Sie Texte gestalten

Der Amiga-Notizblock (also das Programm *Notepad*) bietet aber noch mehr Möglichkeiten, als nur Text einzugeben, zu löschen und umzustellen. Es steht auch eine Vielzahl verschiedener

Methoden zur Wahl, wie Sie Texte gestalten oder auch bestimmte Textpassagen betonen können. Diese verbergen sich hinter den beiden Menüs *Font* und *Style*.

Sie können zum Beispiel mit dem Menü *Font* (engl. *font = Zeichensatz* oder *Schriftart*) den Zeichensatz ändern, mit dem der von Ihnen eingegebene Text im Fenster erscheint. Dies wirkt, als würden Sie bei einer elektrischen Schreibmaschine den Kugelkopf oder das Typenrad tauschen. Dazu müssen Sie zunächst wissen, daß die Menüpunkte im Menü *Font* »Submenüs« sind. Submenüs sind selbst keine Befehle, sondern bringen ein neues Menü mit Befehlen darin zum Vorschein. In diesem Menü können Sie dann genauso einen Befehl auswählen, wie Sie aus einem »normalen« Menü auswählen, das von der Menüleiste herunterklappt.

Probieren Sie es aus: Drücken Sie die Menütaste (die **rechte** Maustaste) und fahren Sie mit der Spitze des Mauszeigers über den Menütitel *Font*. Fahren Sie dann im Menü herunter, bis der Menüpunkt *topaz* hervorgehoben wird. (Die Menü-Punkte im *Font*-Menü tragen alle die Namen von Edelsteinen oder Halbedelsteinen.) Rechts daneben taucht jetzt ein Menü mit den Zahlen 8, 9 und 11 auf. Dies sind die Buchstabengrößen, in denen der Zeichensatz *topaz* zur Verfügung steht. Sowohl der Menüpunkt *topaz* als auch die Zahl 8 sind mit einem Haken versehen. Diese Art der Markierung von Menüpunkten kennen Sie ja bereits. Sie besagt, daß im Augenblick der Zeichensatz *topaz* in der Größe 8 verwendet wird. Fahren Sie nun im Hauptmenü auf und ab. Jedesmal, wenn ein anderer Zeichensatzname hervorgehoben wird, taucht ein anderes Submenü daneben auf. Einige Zeichensätze (zum Beispiel *saphire*) können in bis zu vier Größen ausgegeben werden; andere gibt es nur in zwei Größen.

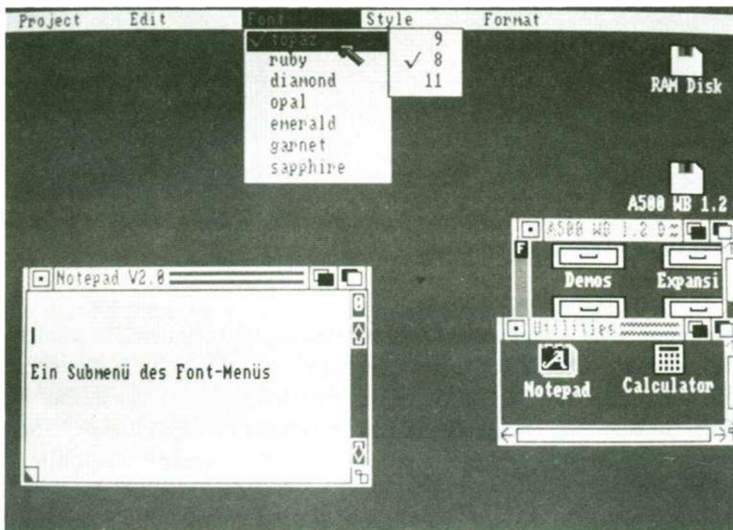


***Bild 4.11:** Ein Submenü des Font-Menüs*

Heben Sie nun den Namen *emerald* hervor und gehen Sie mit dem Mauszeiger nach rechts. Heben Sie im Submenü die Zahl 20 hervor und lassen Sie dann die Menütaste los. Das Diskettenlaufwerk beginnt nun zu arbeiten (wie die Buchstaben eines Zeichensatzes aussehen, wird auf der Workbench-Diskette gespeichert) und danach ändert sich der Inhalt des *Notepad*-Fensters radikal. Der Text, den Sie zuvor eingegeben haben, erscheint jetzt in sehr großen, geschwungenen Lettern. Probieren Sie auch einmal andere Schriftarten und Schriftgrößen aus. Achten Sie aber darauf, daß es nicht ausreicht, nur den Namen der Schrift im Hauptmenü hervorzuheben und dann die Menütaste loszulassen. Das bewirkt nichts. Sie müssen immer auch eine Schriftgröße im Submenü neben dem Schriftnamen hervorheben und dürfen erst dann die Menütaste loslassen – auch wenn Sie nur die Schriftart ändern wollen.

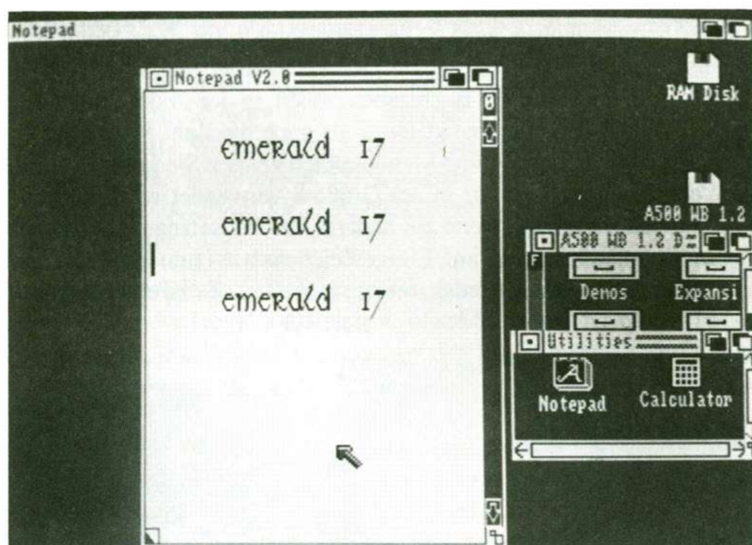


***Bild 4.12:** Text in der Schriftart emerald*

Im Menü *Style* (engl. *style = Zeichenstil* oder *Auszeichnung*) können Sie wählen, ob der Text normal (*Plain*), kursiv (*Italic*), fett (*Bold*) oder unterstrichen (*Underline*) ausgegeben werden soll. Im Gegensatz zur Einstellung für die Schriftart (Zeichensatz), die immer für den ganzen Text gilt, können Sie auf einem Notizblatt durchaus Textpassagen haben, die einen unterschiedlichen Zeichenstil haben. Es können auch mehrere Stilmittel miteinander kombiniert werden. So sind Passagen möglich, die fett gedruckt sind, andere, die kursiv und wieder andere, die unterstrichen und kursiv gedruckt sind. (Im folgenden Abschnitt werden Sie sehen, daß es auch möglich ist, mehrere verschiedene Schriftarten auf einem Notizblatt zu verwenden.)

Probieren Sie es aus: Geben Sie ein paar Zeilen ein, falls Sie das noch nicht getan haben, und setzen Sie die Schreibmarke mitten in den Text. Wählen Sie dann den Befehl *Bold* aus dem

Style-Menü (oder drücken Sie [*A*]-[B]). Der gesamte Textbereich, von der Schreibmarkenposition bis zum Textende, erscheint dadurch fettgedruckt. Setzen Sie die Schreibmarke nun ein Stückchen weiter und wählen Sie *Italic* aus dem *Style*-Menü. Die Buchstaben bis zum Textende sind nun fett und kursiv, die Buchstaben zwischen der alten und der neuen Position der Schreibmarke nur fett. Setzen Sie die Schreibmarke wieder ein Stück weiter und wählen Sie den Befehl *Plain* (engl. für *einfach*) aus dem *Style*-Menü. Alle Buchstaben hinter der Schreibmarke bis zum Textende sehen nun wieder normal aus. Alle Befehle, die Sie aus dem *Style*-Menü wählen, gelten nämlich immer nur bis zu dem Punkt, an dem Sie den nächsten Stilwechsel befehlen!

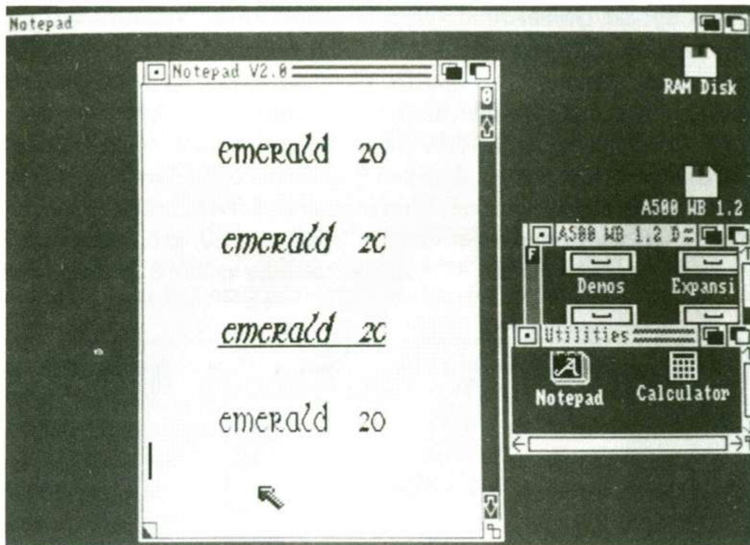


Bild 4.13: *Text mit verschiedenen Stilarten*

4.2.9 Wie Sie den Notizblock nach Ihrem Geschmack gestalten

Das letzte Menü *Format* schließlich bestimmt, wie sich der Notizblock dem Betrachter darbietet. Sie können damit den Notizblock nach Ihrem Geschmack gestalten. Die ersten beiden Menüpunkte bestimmen, in welcher Farbe das »Papier« angezeigt werden soll (*Paper Color*; zu deutsch *Papier-Farbe*) und in welcher Farbe die Buchstaben darauf erscheinen sollen (*Pen Color*; zu deutsch *Stift-Farbe*). Beide Menüpunkte bringen ein Submenü mit vier Farben zum Vorschein, von denen Sie jeweils eine auswählen müssen. (Wie Sie einen Punkt aus einem Submenü auswählen, wissen Sie ja schon aus dem vorangegangenen Abschnitt.) Die Grundeinstellung ist – wie könnte es auch anders sein – schwarzer Text auf weißem Hintergrund. Sie können aber auch weiße Schrift auf weißem Hintergrund wählen – und sehen dann nichts mehr.

Mit dem Menüpunkt *Word Wrap* bestimmen Sie, was passiert, wenn Sie beim Tippen an das Zeilenende beziehungsweise an den rechten Fensterrand kommen. Notepad beginnt dann automatisch, ohne daß Sie auf die [Return]-Taste drücken, eine neue Zeile. Entweder kommt dann der folgende Buchstabe in die neue Zeile, wobei die neue Zeile auch mitten im Wort beginnen kann, oder das Wort, das Sie gerade schreiben, kommt komplett in die nächste Zeile. Ist *Word Wrap* mit einem Haken versehen, werden Zeilen nur zwischen den Wörtern getrennt – es gibt also keine Wörter, die in einer Zeile beginnen und in der nächsten fortgeführt werden. Hat *Word Wrap* jedoch keinen Haken, können neue Zeilen gegebenenfalls auch mitten im Wort beginnen. Das sieht meist häßlich aus und ist wirklich nur für ganz besondere Anwendungen geeignet.

Der nächste Menüpunkt *Global Font* (*globale Schriftart*) ist zunächst mit einem Haken versehen. Dies bewirkt, daß, wie oben beschrieben, jede Änderung der Schriftart über das Menü *Font* immer den gesamten Text betrifft. Wenn Sie jedoch einmal *Global Font* aus dem Menü *Format* auswählen, verschwindet dieser Haken und das *Font*-Menü verhält sich nun wie das *Style*-Menü. Das heißt, wenn Sie eine andere Schriftart aus dem *Font*-Menü wählen, so ändern nur die Buchstaben hinter der aktuellen Schreibmarke ihr Aussehen. Sie können dann also auch erreichen, daß nur ein einzelnes Wort in einer anderen Schriftart erscheint, indem Sie die Schreibmarke davorsetzen, die gewünschte Schriftart und -größe aus dem *Font*-Menü wählen, die Schreibmarke hinter das Wort setzen und eine andere Schriftart wählen.

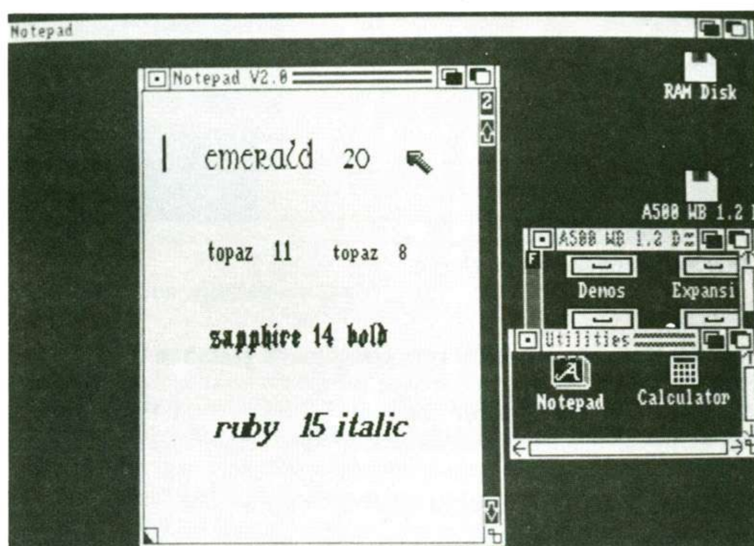


***Bild 4.14:** Text mit verschiedenen Schriftarten*

Die beiden letzten Befehle im *Format*-Menü dienen dazu, die Formatänderungen, die Sie mit dem *Font*- und dem *Style*-Menü durchgeführt haben, wieder rückgängig zu machen. Wenn Sie

Remove Fonts (zu deutsch etwa *Schriftarten entfernen*) aus dem *Format*-Menü wählen, erscheint der gesamte Text wieder in einer Schriftart. Alle Schriftartenwechsel, die Sie vorgenommen haben, werden entfernt und statt dessen erscheinen alle Buchstaben in derselben Schriftart wie der erste Buchstabe des Blattes. Diesen Befehl dürfen Sie aber nicht mit *Global Font* verwechseln. Nach dem Aufruf von *Remove Fonts* können Sie wieder von neuem beginnen, in einzelnen Passagen eines Textes andere Schriftarten zu verwenden. Ist hingegen *Global Font* abgehakt, erscheinen immer alle Buchstaben des Textes in der zuletzt gewählten Schriftart.

Genauso wie *Remove Fonts* auf die Schriftarten, wirkt *Remove Styles* (zu deutsch etwa *Stilwechsel entfernen*) auf die mit dem Style-Menü festgelegten Schriftstile. Nach dem Aufruf von *Remove Styles* erscheint der gesamte Text *Plain* (also nicht fett, nicht kursiv und nicht unterstrichen).

4.3 Wichtige Eigenschaften von Projekten

Nun haben Sie bereits die Grundzüge eines ziemlich leistungsfähigen und komplexen Werkzeugs kennengelernt (eine echte Textverarbeitung ist allerdings noch um einiges komplizierter). Worum es jedoch eigentlich in diesem Abschnitt geht, ist bislang noch zu kurz gekommen: die Projekte beziehungsweise Dokumente. Der Notizblock ist ja – anders als die Uhr zum Beispiel – ein Programm, mit dem Sie wirklich ein Dokument bearbeiten. Den Text, den Sie im Notizblock erstellt haben, verlieren Sie normalerweise in dem Augenblick, in dem Sie das Fenster des Programms *Notepad* wieder schließen – wenn Sie ihn nicht sichern. Ein Projekt (wie der Inhalt des Notizblocks) kann auf der Diskette abgespeichert, wieder von Diskette gelesen und ausgedruckt werden. Die meisten Programme – so auch *Notepad* – besitzen für diese Operationen mit Projekten ein spezielles Menü: das Project-Menü.

4.3.1 Das Project-Menü

Das erste Menü ganz links in der Menüleiste von *Notepad* trägt den Titel *Project*. Es dient zur Auswahl von Operationen, die den Text als Ganzes, also als ein Projekt, betreffen. Dieses Menü ist in gleicher oder ähnlicher Form bei vielen anderen Programmen auch zu finden. Es lohnt sich deshalb für Sie, sich ein wenig näher damit zu beschäftigen.

Der erste Befehl, *New* (engl. für *neu*), löscht das *Notepad*-Fenster und beginnt so ein neues Projekt. Aller Text, den Sie eingegeben haben, geht dadurch verloren. Vermeiden Sie es also nach Möglichkeit, diesen Befehl aus Versehen auszuwählen.

Der Befehl *Save* (engl. *to save = sichere*) dient zum Abspeichern des gerade erstellten Textes auf Diskette. Wenn Sie den Text das erste Mal mit diesem Befehl abspeichern, müssen Sie ihm dazu natürlich einen Namen geben, wofür extra ein Requester erscheint.

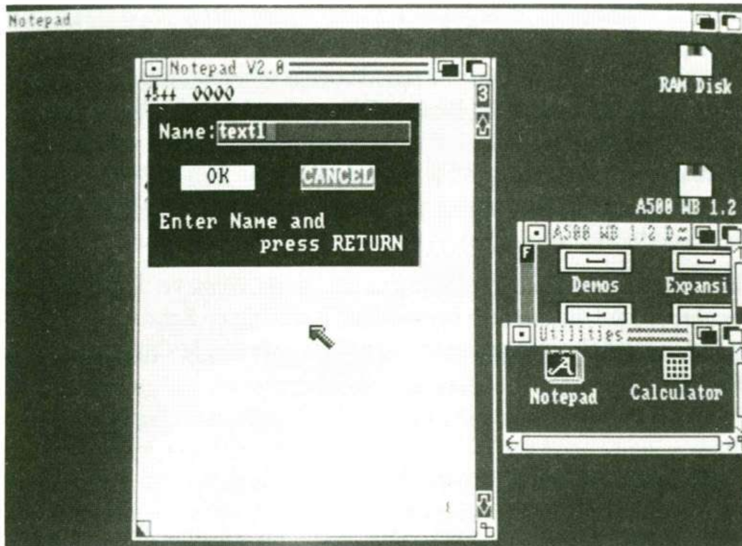


***Bild 4.15:** Der Save-Requester*

Der *Save*-Requester besteht aus einem Text-Gadget, in das Sie einen Namen eingeben können, und zwei Knöpfen. Klicken Sie das Text-Gadget neben der Bezeichnung *Name* an und tippen Sie den gewünschten Namen des Dokuments (Projekts) ein. Schließen Sie den Namen (wie jede Texteingabe) mit der [RETURN]-Taste ab und betätigen Sie dann (mit der Maus) den Knopf OK. Ihr Text wird nun unter dem eingegebenen Namen auf die Diskette geschrieben und wird somit »gesichert«. Im Gegensatz zu den Daten im RAM-Speicher des Amiga überstehen die Daten auf der Diskette zum Beispiel das Ein- und Ausschalten des Geräts und sind somit verhältnismäßig sicher zu nennen. Falls Sie es sich anders überlegen, wenn der *Save*-Requester erschienen ist, und den Text nicht mehr abspeichern wollen, brauchen Sie nur auf den Knopf CANCEL zu klicken. Der Requester verschwindet dann und nichts weiter geschieht – insbesondere wird nichts auf der Diskette abgespeichert.

Hatten Sie den Text schon einmal gesichert, so hat er schon einen Namen und es wird einfach noch einmal unter dem alten Namen abgespeichert, wenn Sie *Save* aus dem *Project*-Menü wählen. Dabei wird die letzte gespeicherte Version allerdings zerstört!

Der Befehl *Save As* (engl. für *Sichere als/unter dem Namen*) wirkt genau wie *Save*, nur, daß Sie immer einen Namen für den zu sichernden Text eingeben müssen – auch dann, wenn Sie ihn schon einmal gesichert hatten und ihm dabei einen Namen gegeben haben. *Save As* dient vor allem zum Ablegen eines schon einmal abgespeicherten Textes unter einem neuen Namen.

Save und *Save As* sind (auch in den meisten anderen Werkzeugen) die Befehle, die neue Projekte erzeugen. Sie können das leicht überprüfen: Starten Sie *Notepad*, falls Sie das noch nicht getan haben, und geben Sie einen kurzen Text auf dem ersten Blatt ein. Sichern Sie diesen

dann mit *Save* unter einem beliebigen Namen auf der Diskette. Daraufhin erscheint nach kurzer Zeit im Fenster *Utilities* (in dem ja das Werkzeug *Notepad* liegt) ein neues Piktogramm, das wie ein Blatt Papier aussieht und unter dem der Name steht, den Sie soeben Ihrem Text gegeben haben. Dies ist ein Projekt, wie es *Notepad* erzeugen und bearbeiten kann.

Mit dem Befehl *Open* können Sie ein solches Projekt wieder im *Notepad*-Fenster anzeigen lassen und bearbeiten, wenn Sie es einmal abgespeichert haben. Nachdem Sie *Open* ausgewählt haben, erscheint wieder der Requester, den Sie ja schon von den Befehlen *Save* und *Save As* her kennen, und Sie müssen einen Namen eingeben. Drücken Sie danach auf [RETURN] und dann mit der Maus auf OK und der Text mit den angegebenen Namen wird eingelesen und angezeigt. Dabei verschwindet aber der alte Text, den Sie zuvor im *Notepad* bearbeitet hatten. Wenn Sie ihn nicht zuvor gesichert hatten, ist er verloren.

Mit den beiden Menüpunkten *Print* (engl. *to print = drucken*) und *Print As* (engl. für *drucke als*) können Sie den Inhalt des Notizblocks ausdrucken. Beide Menüpunkte bringen Submenüs (siehe oben) zum Vorschein, die bestimmen, wie der Ausdruck stattfindet beziehungsweise aussieht.

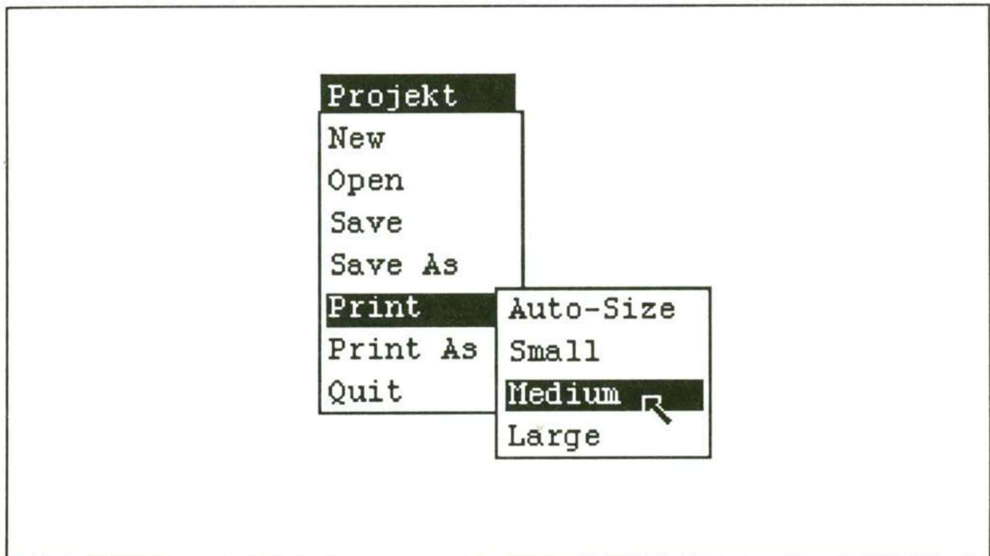


Bild 4.16: *Das Submenü Print*

Zunächst müssen Sie im Submenü *Print As* auswählen, ob beim Ausdruck versucht werden soll, den Text so erscheinen zu lassen, wie er auch auf dem Bildschirm erscheint. Sie haben hier die Auswahl zwischen *Graphic* (engl. für *Grafik*) und *Draft* (engl. für *Entwurf*). Diese beiden Punkte des Submenüs sind Schalter, die sich gegenseitig ausschließen. (Solche Menü-Schalter sind Ihnen ja bereits bekannt.) Ist der Schalter *Graphic* an, so werden bei einem Ausdruck des

Textes alle Besonderheiten und speziellen Effekte, die in den vorangegangenen Abschnitten vorgestellt wurden, auf dem Papier ausgegeben. Hierzu gehören zum Beispiel die verschiedenen Schriftarten und Schriftstile. Wenn Sie einen Mehrfarbdrucker verwenden, werden auch die auf dem Bildschirm zu sehenden Farben beim Ausdruck verwendet. Sie erhalten dann einen Ausdruck, der dem entspricht, was Sie auch am Bildschirm sehen.

Das funktioniert natürlich nur dann, wenn Sie einen Drucker verwenden, der grafikfähig ist. Wenn Sie einen nicht grafikfähigen Drucker anschließen oder den Schalter *Draft* im Submenü *Print As* anschalten, so werden zwar alle Zeichen des Textes korrekt ausgegeben, die verschiedenen Schriftarten und Zeichenstile und Farben werden ignoriert. Der letzte Punkt im Submenü *Print As* ist nur dann wichtig, wenn Sie einen sehr langen Text ausdrucken – wozu der Notizblock eigentlich nicht geeignet ist. Wenn Sie *Form-Feeds* (engl. für *Seitenvorschübe*) auswählen und damit auch abhaken, werden beim Ausdruck am oberen und unteren Rand des Papiers einige Zeilen frei gelassen. Ist *Form-Feeds* nicht abgehakt, wird es bei entsprechend langen Texten vorkommen, daß der Drucker über das Seitenende hinweg druckt. Das kann bei der Verwendung von Endlospapier sinnvoll sein, ist bei einzelnen Blättern aber sicherlich nicht erwünscht.

Nachdem Sie mit *Print As* diese Ausdruckoptionen gewählt haben, können Sie mit *Print* den Druckvorgang starten. *Print* ist ebenfalls ein Submenü mit den Unterpunkten *Auto-Size* (engl. für *automatische Größe*), *Small* (engl. für *klein*), *Medium* (engl. für *mittel*) und *Large* (engl. für *groß*). Alle diese Befehle bewirken einen Ausdruck des Notizblockinhalts auf dem Drucker (sofern Sie einen besitzen und korrekt angeschlossen haben). Wählen Sie *Auto-Size*, entscheidet das Programm, in welcher Größe Ihr Text ausgegeben wird. Bei *Small*, *Medium* und *Large* bestimmen Sie selbst die Größe des Ausdrucks (klein, mittel oder groß). Experimentieren Sie am besten selbst ein wenig mit den verschiedenen Möglichkeiten herum! Es kostet ein wenig Papier, ist aber die beste Methode, um zu wirklich (für Sie) brauchbaren Ergebnissen zu kommen.

Der letzte Befehl im *Project*-Menü ist ein besonders schwerwiegender. Mit *Quit* verlassen Sie das Werkzeug *Notepad*, und sämtlicher Text, den Sie nicht gesichert haben, ist verloren. Denselben Effekt wie mit *Quit* können Sie aber auch erreichen, indem Sie das Schließ-Gadget des *Notepad*-Fensters betätigen.

4.3.2 Das Info-Fenster eines Werkzeugs

Nachdem Sie jetzt ja die typische Arbeitsweise eines Werkzeugs andeutungsweise kennengelernt haben, sollten Sie sich als nächstes einmal anschauen, was die Workbench an Informationen über Projekte und Werkzeuge zu vermelden hat. Hierzu ist wie üblich der *Info*-Befehl der Workbench zuständig. Verlassen Sie deshalb nun den Notizblock mit dem Befehl *Quit* oder mit Hilfe des Schließ-Gadgets. Selektieren Sie dann das *Notepad*-Piktogramm (mit **einem** Mausklick darauf) und wählen Sie den Befehl *Info* aus dem *Workbench*-Menü.

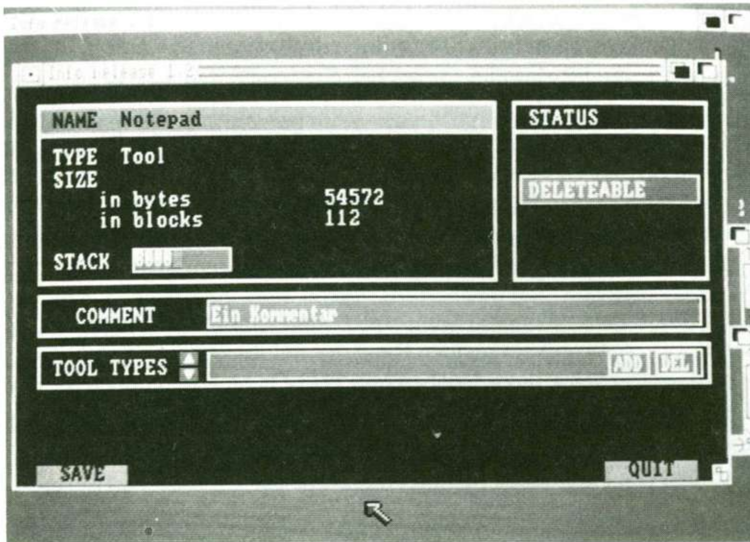


Bild 4 .17: Das Info-Fenster eines Werkzeugs

Dieses Fenster dürfte Ihnen bereits von den anderen Workbench-Objekten her vertraut sein. Es enthält nur einige zusätzliche Komponenten. Links in diesem Fenster steht unter dem Namen der Objekt-Typ, in diesem Fall *Tool* (engl. für *Werkzeug*). Darunter steht die Größe in Byte (Buchstaben) und Blöcken (die jeweils 512 Byte enthalten). Ähnliche Angaben gibt es ja auch bei Disketten-Infos. Darunter befindet sich nun außerdem ein neues Feld namens Stack. Hinter dem Wort Stack können Sie eine Zahl eingeben, die bestimmt, wieviel einer besonderen Sorte Speicherplatz das Werkzeug beim Start im RAM des Amiga reserviert bekommt. Diese Angabe ist für Sie erst dann interessant, wenn Sie ein wenig mehr von den Innereien des Amiga kennengelernt haben; sie sei aber der Vollständigkeit halber erwähnt. (Falls Sie bei der Anwendung eines Programms einmal eine Fehlermeldung bekommen, in der der Begriff *stack overflow* vorkommt, können Sie versuchen, diesen Wert zu ändern (also zu vergrößern). Normalerweise beträgt er 4000 (Byte), wird aber nicht angezeigt. Tragen Sie schrittweise größere Werte ein, und versuchen Sie den Vorgang, der zum Fehler führte, noch einmal.)

Die anderen Felder im Infofenster sollten Ihnen alle bereits vom Info-Fenster einer Schublade oder Diskette bekannt sein. Hinter *Comment* können Sie zum Beispiel eine Bemerkung (Kommentar) zu diesem Werkzeug eintragen und im Feld *Status* können Sie ein Werkzeug vor unbeabsichtigtem Löschen schützen (indem Sie den »Schalter« DELETEABLE darin auf NOT DELETEABLE stellen).

Interessant ist aber wieder das Feld TOOL TYPES, das es nur im Info-Fenster von Werkzeugen gibt. Sie können hier einen oder mehrere Texte eintragen (wie, erfahren Sie gleich), die dem Programm beim Start mitgeteilt werden. Diese Texte nennt man »Parameter« und Sie stellen

praktisch Befehle an das Programm dar – nur, daß diese Befehle nicht mit der Maus erteilt werden, sondern dem Programm beim Start automatisch mitgeteilt werden. Indem Sie hier im Info-Fenster diese Parameter eintragen, erreichen Sie ferner, daß Sie sie nur einmal angeben müssen und daß sie trotzdem bei jedem Start wirksam werden. *Notepad* können Sie zum Beispiel mit Hilfe solcher Parameter mitteilen, wie groß das Fenster des Programms beim Start sein soll, welche Schriftart verwendet werden soll, solange Sie nichts anderes angeben, und so weiter. Typischerweise dienen Parameter dazu, die Grundeinstellungen (der Fachmann sagt *Default-Einstellungen* oder *Defaults* dazu) zu verändern, die in einem Programm nach dem Start gelten.

Wenn Sie einen neuen Parameter eintragen wollen, müssen Sie zunächst auf das ADD (engl. *to add = hinzufügen/ ergänzen*) am rechten Rand des Feldes TOOL TYPES klicken. Klicken Sie dann in das blaue Rechteck links daneben (ein Text-Gadget) und tippen Sie den Parameter ein. Ein solcher Parameter besteht immer aus einem Wort in Großbuchstaben, gefolgt von einem Gleichheitszeichen, hinter dem weitere Informationen folgen, deren Aussehen abhängig vom Namen des Parameters (dem ersten Wort) ist.

Wenn Sie mehr als einen Parameter eintragen wollen, brauchen Sie einfach nur erneut auf ADD zu klicken und das Textgadget links daneben wird wieder frei. Sie können dann den Text des nächsten Parameters eintippen. Sie können sich auch alle Parameter, die Sie oder ein anderer schon eingegeben haben, nachträglich anschauen oder auch ändern. Hierzu dienen die beiden Pfeile rechts neben der Beschriftung TOOL TYPES. Indem Sie diese anklicken, können Sie zwischen den Parametern vor- und zurückblättern. Wenn ein Parameter erst einmal in dem blauen Rechteck neben den beiden Pfeilen erschienen ist, können Sie ihn auch ändern. Klicken Sie dazu einfach mit der Maus auf eine beliebige Stelle im Text des Parameters. Sie können ihn dann mit der [Del]-, der [Backspace]- und den Cursor-Tasten abändern, wie Sie wünschen. Ihnen stehen dazu dieselben Möglichkeiten zur Verfügung wie in jedem Text-Gadget (diese Möglichkeiten werden vollständig in Kapitel 6 beschrieben). Wenn Sie einen Parameter nicht nur ändern, sondern komplett löschen wollen, brauchen Sie nur auf den Knopf DEL (für *delete = lösche*) am rechten Rand des Feldes TOOL TYPES zu klicken. Der Parameter verschwindet daraufhin und der vorangehende (falls es einen gibt) erscheint am Bildschirm.

Dies war zunächst einmal nur eine grundsätzliche Beschreibung der Funktion des Feldes TOOL TYPES im Info-Fenster eines Projekts. Eine vollständige Liste der Parameter, die Sie beim Programm *Notepad* verwenden können, erhalten Sie im folgenden Abschnitt.

Wie bei allen anderen Info-Fenstern gilt auch hier beim Info-Fenster eines Projekts übrigens, daß die Änderungen nur dann gültig werden, wenn Sie das Fenster mit einem Klick auf den Knopf SAVE schließen. Schließen Sie es mit QUIT oder mit einem Klick ins Schließ-Gadget, werden alle Änderungen zwischen dem Öffnen und Schließen des Fensters verworfen. Bei Parametern kommt noch dazu, daß fast alle Änderungen erst beim nächsten Start des entsprechenden Programms (Öffnen des Projekt-Piktogramms) Wirkung zeigen. Eine Änderung des Inhaltes von TOOL TYPES wirkt niemals auf vorher gestartete, schon laufende Programme.

4.3.3 Die Parameter von Notepad

Nachdem Sie nun wissen, wie Sie einem Programm über das Info-Fenster Befehle (Parameter) zukommen können lassen, sollen Sie auch gleich die vollständige Liste der Parameter, die *Notepad* versteht, erhalten. Sie können diese neue Möglichkeit dann sofort nach Herzenslust ausprobieren.

Der Parameter WINDOW bestimmt die Größe des Notepad-Fensters direkt nach dem Start des Programms. Hinter dem Gleichheitszeichen (=) müssen bei diesem Parameter vier von Kommata getrennte dreistellige Zahlen stehen. Diese Zahlen geben nacheinander an, welchen Abstand der linke Fensterrand vom linken Bildschirmrand, welchen Abstand der obere Fensterrand vom oberen Bildschirmrand, welche Breite und welche Höhe das Fenster hat. (Alle diese Angaben müssen dreistellig sein und werden in Bildschirmpunkten oder *Pixeln* gemessen.) Wenn Sie also wollen, daß das Notepad-Fenster beim Start 300 Punkte breit und 100 Punkte hoch ist und die linke obere Ecke 10 Punkte vom linken und 40 Punkte vom oberen Bildschirmrand entfernt ist, müssen Sie »WINDOW=010,040,300,100« im Feld TOOL TYPES eintragen. (Beachten Sie die führende Null bei 010 und 040. Diese Null oder eine Leerstelle müssen Sie eingeben, wenn eine Zahl weniger als drei Stellen hat!)

Der Parameter FONT bestimmt, in welcher Schriftart die Buchstaben, die Sie eingeben, erscheinen, solange Sie keine Schriftart aus dem *Font*-Menü wählen (also die Default-Schriftart). Hinter dem Gleichheitszeichen müssen Sie den Namen der gewünschten Schriftart (derselbe wie im Font-Menü), einen Punkt und die gewünschte Größe angeben. Wenn Sie also wollen, daß sofort nach dem Start eingegebene Buchstaben in der Schriftart *diamond* und der Größe 20 erscheinen, müssen Sie »FONT=diamond.20« im Feld TOOL TYPES eintragen.

Der Parameter FLAGS erlaubt es Ihnen, einige der als Schalter wirkenden Menüpunkte im *Project*- und *Format*-Menü schon vor dem Start zu bestimmen. So können Sie im Submenü *Print As* zum Beispiel festlegen, ob Sie beim Drucken nur den »nackten Text« ohne Schriftart- und Stilwechsel (*Draft*) oder den Text, so wie er auf dem Bildschirm erscheint (*Graphic*), ausgeben wollen und ob Sie am Seitenende ein paar Leerzeilen wünschen. Voreingestellt ist nach dem Start *Graphic*; die »Schalter« *Draft* und *Form-Feeds* sind aus. Indem Sie »FLAGS=draft« im Feld TOOL TYPES eintragen, können Sie dafür sorgen, daß immer im *Draft*-Modus gedruckt wird (sofern Sie nicht explizit noch einmal *Graphic* wählen) und mit dem Parameter »FLAGS=formfeed« können Sie genauso den Schalter *Form-Feeds* aktivieren.

Nach dem Start von Notepad ist der Schalter *Global Font* im Menü *Format* zunächst an (abgehakt). Jede Auswahl einer neuen Schriftart aus dem *Font*-Menü betrifft dann immer den gesamten Text. Wenn Sie jedoch im Feld TOOL TYPES des Info-Fensters von *Notepad* »FLAGS=local« eintragen, ist *Global Font* gleich nach dem Start von *Notepad* aus und Sie können mehrere verschiedene Schriftarten auf einem Blatt verwenden. Mit »FLAGS=global« können Sie in ähnlicher Form *Global Font* wieder aktivieren. Genauso können Sie den Schalter *Word wrap* im Menü *Format* beeinflussen. »FLAGS=nowrap« macht *Word wrap* inaktiv.

Und schließlich können Sie mit dem Parameter »FLAGS=nofonts« verhindern, daß *Notepad* beim Start auf der Workbench-Diskette nachsieht, welche Schriftarten vorhanden sind. Der

Start von *Notepad* geht dann wesentlich schneller und das *Font*-Menü bleibt fast leer. Falls Sie später doch mehrere Schriftarten verwenden wollen, können Sie das *Font*-Menü mit dem Befehl *Read Fonts* (engl. für *Schriftarten einlesen*) aus dem *Projekt*-Menü nachträglich aufbauen.

Wenn Sie eine frühe Version des Programms *Notepad* besitzen, beziehungsweise zusammen mit Ihrem Amiga 500 erhalten haben, kann es übrigens sein, daß einige der hier beschriebenen Parameter (insbesondere eine Kombination mehrerer FLAGS-Parameter) nicht funktionieren. Bei der von mir getesteten *Notepad*-Version war dies der Fall. Die Firma Commodore versichert jedoch, daß die endgültige Version von *Notepad* alle hier beschriebenen Parameter erkennen wird.

4.4 Zusammenhänge zwischen Werkzeugen und Projekten

Projekte und Werkzeuge gehören stets eng zusammen. Oft gibt es zu jedem Projekt nur genau ein Werkzeug, mit dem es bearbeitet werden kann, und alle Projekte, die mit einem bestimmten Werkzeug bearbeitet werden können, besitzen auch dasselbe Piktogramm.

4.4.1 Öffnen eines Projektes

Der Zusammenhang zwischen Projekten und Werkzeugen wird aber nicht nur durch solche Äußerlichkeiten bestimmt. Sie haben stattdessen auch die Möglichkeit, ein Projekt zu öffnen wie ein Werkzeug. Das heißt, wenn Sie durch einen Doppelklick oder den Befehl *Open* ein Projekt statt eines Werkzeuges öffnen, so wird Ihnen – wie von anderen Objekten bekannt – gezeigt, was es enthält. Im Falle eines Textes muß also dieser Text in einem Fenster gezeigt werden. Hierzu wird das Projekt natürlich ein Werkzeug benötigen, das das Projekt (in diesem Fall den Text) dann anzeigt. Idealerweise sollte das das Werkzeug sein, mit dem das Projekt erstellt wurde, denn nur dieses Werkzeug »weiß« wahrscheinlich, wie man das Projekt richtig darstellt.

Wenn Sie ein Projekt öffnen, prüft die Workbench deshalb zunächst, zu welchem Werkzeug das Projekt gehört, und startet dieses. Gleichzeitig wird dem Werkzeug mitgeteilt, welches Projekt der Anwender geöffnet hat. (Dies geschieht auf ähnlichem Wege wie die Mitteilung der oben beschriebenen Parameter.) Das Werkzeug liest dieses Projekt daraufhin ein und zeigt es am Bildschirm. Dies ist wesentlich praktischer, als immer zuerst ein Werkzeug starten zu müssen und dann das Projekt einzulesen, wenn man es bearbeiten oder nur betrachten will. Hier geschieht also beim Öffnen eines Projektes genau das, was Sie aus Ihrer bisherigen Erfahrung mit anderen Objekten erwarten können.

4.4.2 Das Info-Fenster eines Projektes

Die Verbindung zwischen einem Projekt und dem zugehörigen Werkzeug wird auch dann sichtbar, wenn man das Info-Fenster des Projektes öffnet (also dem Projekt den Befehl *Info* erteilt).

Um das auszuprobieren, benötigen Sie ein Projekt. Falls Sie bei Ihren Experimenten mit dem Notizblock bisher kein Projekt erzeugt haben, sollten Sie das deshalb nun tun. Starten Sie dazu *Notepad*, tippen Sie ein paar Sätze ein und sichern Sie diesen Text dann (mit dem oben beschriebenen Befehl *Save*) unter einem beliebigen Namen ab. Sie können das *Notepad*-Fenster dann schließen. Unter diesem Namen erscheint das Projekt dann als Piktogramm im Fenster *Utilities*, in dem auch *Notepad* liegt. (Falls das Piktogramm nicht sofort erscheint, schließen Sie das Fenster *Utilities* kurz und öffnen Sie es erneut.) Selektieren Sie nun dieses neue Piktogramm und wählen Sie den Befehl *Info* aus dem Menü *Workbench*.

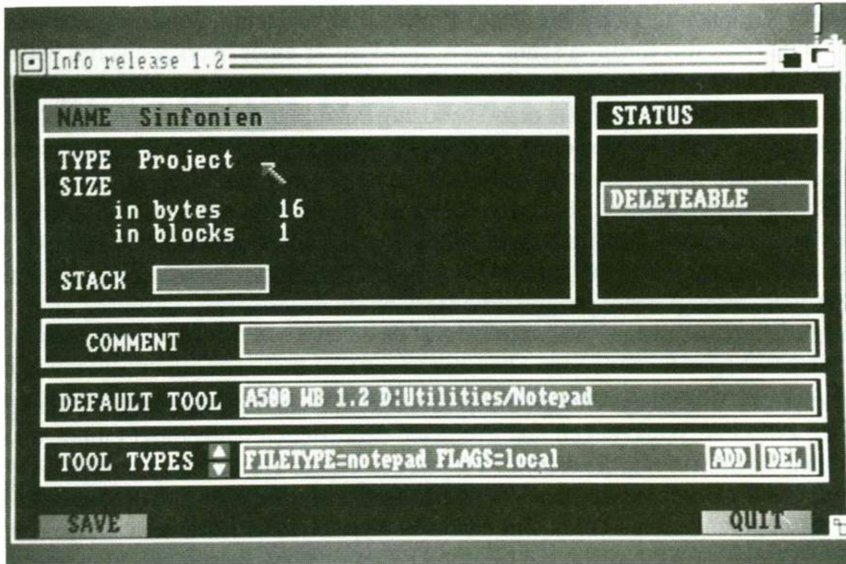


Bild 4.18: *Das Info-Fenster eines Projektes*

Das Info-Fenster eines Projektes sieht fast genauso aus wie das eines Werkzeugs. Es enthält nur ein zusätzliches Text-Gadget namens DEFAULT TOOL. Dieses Text-Gadget stellt die Verbindung zwischen einem Projekt und dem zugehörigen Werkzeug her. Im Text-Gadget DEFAULT TOOL kann nämlich der Name eines Werkzeuges eingegeben werden, das aufgerufen werden soll, wenn das Projekt geöffnet wird. Beim ersten Ablegen eines neuen Projektes auf der Diskette hinterlegt das erzeugende Programm hier meist seinen eigenen Namen.

Betrachten Sie zum Beispiel das Info-Fenster eines *Notepad*-Projektes, so steht dort zum Beispiel »A500 WB 1.2 D:Utilities/Notepad«. »A500 WB 1.2 D:« bedeutet, daß das Werkzeug auf der Workbench-Diskette zu finden ist (diese hat bei Ihnen vielleicht auch einen anderen Namen), »Utilities/«, daß es in der Schublade Utilities liegt und »Notepad« schließlich ist natürlich der Name des Werkzeugs selbst. (Mehr zu dieser ungewöhnlichen Schreibweise

eines Dateinamens beziehungsweise der Disketten und Schubladen, in denen er sich befindet, können Sie erfahren, wenn Sie das Kapitel *Dateien und Dateiverzeichnisse* lesen.) Sie können hinter DEFAULT TOOL theoretisch einen anderen Namen eintragen und diesen dann abspeichern, indem Sie auf SAVE klicken. Wenn Sie zum Beispiel einmal eine leistungsfähigere Textverarbeitung erwerben, können Sie auf diese Weise erreichen, daß beim Öffnen von Texten, die mit *Notepad* erstellt wurden, dieses neue Werkzeug gestartet wird und nicht mehr *Notepad*.

4.4.3 Parameter eines Projektes

Eine andere Verbindung zwischen einem Projekt und »seinem« Werkzeug stellt das Feld TOOL TYPES dar, das Ihnen ja schon vom Info-Fenster eines Werkzeugs her bekannt ist. Wenn Sie bei einem Werkzeug in diesem Feld einen oder mehrere Befehle (Parameter) eintragen, so werden diese Befehle dem Werkzeug beim Start mitgeteilt. Genauso verhält es sich, wenn Sie im Info-Fenster eines Projektes hinter TOOL TYPES Parameter eintragen. Auch diese werden dem Werkzeug mitgeteilt, wenn Sie es – natürlich »indirekt« über das Projekt-Piktogramm – starten. Sie können bei dem Piktogramm eines mit *Notepad* erstellten Projektes also dieselben Parameter verwenden wie beim Piktogramm von *Notepad* selbst. Diese werden aber nur dann gültig, wenn Sie *Notepad* über dieses Projekt-Piktogramm öffnen und nicht dann, wenn Sie *Notepad* direkt oder über ein anderes Projekt-Piktogramm öffnen.

Am Beispiel des Werkzeugs *Notepad* und seiner *Projekte* habe ich versucht, die wichtige Beziehung zwischen Werkzeugen und Projekten darzustellen. Unbedingt einprägen sollten Sie sich auch die Bedeutung, die das Info-Fenster sowohl für Werkzeuge wie auch für Projekte hat. Damit soll die Beschreibung des Verhältnisses von Werkzeugen und Projekten aber auch erst einmal abgeschlossen werden. Im nächsten Kapitel werden Sie von einigen weiteren wichtigen Aspekten erfahren, die Sie bei der Arbeit mit Werkzeugen beachten müssen, und schließlich auch noch die Werkzeuge kennenlernen, die sich außer der Uhr, dem Rechner und dem Notizblock noch auf der Workbench-Diskette befinden. Sie werden diese Werkzeuge bei der Arbeit mit dem Amiga vielleicht nicht täglich, aber doch recht häufig benötigen.

5 Werkzeuge der Workbench

Auf der Workbench-Diskette, die Sie beim Kauf des Amiga 500 erhalten, befinden sich neben Uhr (*Clock*), Rechner (*Calculator*) und Notizblock (*Notepad*) noch eine ganze Reihe weiterer Werkzeuge. Mit diesen Werkzeugen wird sich dieses Kapitel schwerpunktmäßig befassen. Zunächst aber geht es um einen Aspekt von Werkzeugen, der bislang noch nicht genügend gewürdigt wurde, obwohl er erstens sehr wichtig ist und zweitens den Amiga 500 deutlich von anderen einfacheren Heimcomputern unterscheidet: das Multitasking.

5.1 Wie man sich zerreißt: Multitasking

Wer hat nicht schon einmal den Stoßseufzer gehört oder selbst ausgesprochen: »Ich kann mich doch nicht zerreißen«? Das soll im allgemeinen bedeuten, daß jemand aufgefordert wird, mehrere Aufgaben gleichzeitig zu erledigen, was natürlich unmöglich ist – zumindest für einen Menschen. Der Amiga kann sich zerreißen. Oder anders formuliert: Der Amiga 500 kann mehrere Programme (scheinbar) gleichzeitig bearbeiten. Sie haben diese Fähigkeit auch schon in Aktion gesehen – nur vielleicht nicht näher darüber nachgedacht. Nachdem man ein Programm wie *Notepad* gestartet hat und dieses Programm sein Werkzeug geöffnet hat, hält einen nämlich niemand davon ab, in ein anderes Fenster zu klicken und dort mit einem anderen Programm zu arbeiten.

Das beste Beispiel für dieses Verhalten ist die Workbench selbst, die ja auch ein Programm ist. Nachdem von ihr aus ein Werkzeug geöffnet wurde, kann man problemlos weitere Schubladen öffnen, verschieben oder auch weitere Werkzeuge öffnen. Während der ganzen Zeit läuft ein erst einmal gestartetes Programm (Werkzeug) weiter, wie zum Beispiel auch bei der Uhr sehr gut zu sehen ist, die ja auch ein Werkzeug ist.

Auf anderen Computern, von denen Sie möglicherweise auch den einen oder anderen kennen, ist ein solches Verhalten durchaus nicht selbstverständlich. Startet man dort ein Programm, so

»übernimmt es« den ganzen Computer, beansprucht den ganzen Speicher für sich, löscht unter Umständen alles, was sich darin befindet und beansprucht auch den gesamten Bildschirm für sich. Auf dem Amiga üben die Programme hingegen (weitgehend) friedliche Koexistenz, wozu unter anderem auch die Fenster beitragen, die dafür sorgen, daß den Programmen genau abgegrenzte Bereiche des Bildschirms zugeteilt werden können, ohne sie dadurch allzusehr einzuschränken.

Der Fachmann nennt dieses Verhalten (die gleichzeitige Aktivität mehrerer Programme) eines Computers »Multitasking«, weil der Computer gleichzeitig an mehreren Aufgaben (engl. *tasks*) arbeiten kann. Dies ist natürlich nur eine scheinbare Gleichzeitigkeit. Auch ein Computer kann immer nur ein Programm zur gleichen Zeit bewältigen. Der Amiga arbeitet aber immer nur ein kleines Stück eines Programms ununterbrochen ab. Danach »springt« er zu einem anderen und bearbeitet dieses für einen kleinen Augenblick und so weiter. Er wechselt aber so schnell von Programm zu Programm, daß für den langsamen Menschen der Eindruck entsteht, er würde diese Programme gleichzeitig bearbeiten.

5.1.1 Vorteile des Multitasking

Eventuell werden Sie sich nun fragen, wozu Multitasking denn überhaupt gut sein soll. Denn obwohl der Computer scheinbar schnell genug ist, mehrere Aufgaben gleichzeitig zu bewältigen, kann ja kaum von einem Menschen verlangt werden, blitzschnell zwischen der Bedienung von zwei oder drei Programmen hin und her zu wechseln. Ganz davon abgesehen, daß für einen solchen ständigen Wechsel wohl kaum Bedarf besteht.

Dies ist auch völlig richtig, und auch das Multitasking des Amiga ist nicht dafür gedacht, daß Sie simultan einen Text tippen und Ihre Buchhaltung erledigen. Es gibt aber bei vielen Programmen Vorgänge, die sehr viel Zeit in Anspruch nehmen – einige Minuten oder gar eine Stunde und mehr. Das Ausdrucken langer Dokumente und aufwendige Kalkulationsaufgaben sind zum Beispiel solche Vorgänge. Wenn Ihr Computer immer nur ein Programm bearbeiten kann, müssen Sie immer darauf warten, daß ein Programm fertig wird, bevor Sie mit der Arbeit mit diesem oder einem anderen Programm fortfahren können. Oder Sie müssen sich einen zweiten Computer kaufen. Beherrscht Ihr Computer aber das Multitasking, können Sie, während sich das eine Programm seiner langwierigen (und wahrscheinlich langweiligen) Tätigkeit hingibt, ein anderes Programm für eine andere Aufgabe nutzen.

Aber auch ein schneller Wechsel zwischen verschiedenen Programmen kann hilfreich sein. Angenommen, Sie wollen einen Bericht schreiben und verwenden dazu ein Textverarbeitungsprogramm. Plötzlich merken Sie, daß Ihnen einige Daten fehlen. Diese Daten verwalten Sie normalerweise mit einem Datenbankprogramm. Um sie jetzt zu betrachten, müßten Sie dieses Programm nun starten. Auf der Workbench des Amiga können Sie das sofort tun und in der Datenbank nachsehen, ohne die Textverarbeitung zu verlassen (und den Text abzuspeichern). Danach sind Sie blitzschnell wieder zurück in der Textverarbeitung. Gerade das Starten und Verlassen von Programmen sind nämlich Vorgänge, die auf einem (bei aller Leistungsfähigkeit doch vergleichsweise kleinen) Computer wie dem Amiga 500 recht zeitraubend sein können.

Die Möglichkeit des Multitasking kann Ihnen also in vielerlei Hinsicht Zeit und Mühe ersparen und nutzt zugleich den Computer auch noch viel besser aus. Selbst ein kleiner Computer wie der Amiga 500 ist nämlich immer noch so schnell, daß er einen Großteil seiner Zeit nichts anderes tut, als auf Ihre Eingaben zu warten. In dieser Zeit könnte er genausogut schon sinnvollere Dinge tun – zum Beispiel drucken oder rechnen. Der Amiga 500 tut das, andere Computer nicht!

5.1.2 Das Problem Speicherplatz

Multitasking ist aber auch nicht ganz ohne Probleme. Wenn zwei identische Programme (Werkzeuge) gleichzeitig arbeiten, benötigen sie auch den doppelten Speicherplatz im Computer. Gerade bei großen und leistungsfähigen Programmen werden Sie rasch feststellen, daß selbst die circa 250.000 oder gar 1.040.000 Zeichen, die der Amiga in seinem RAM-Speicher ablegen kann, nicht allzuviele sind. Während Sie auf der Workbench sind, können Sie dies sehr schön beobachten. Der noch freie Speicherplatz wird dann nämlich in der Titelleiste des Bildschirms immer angezeigt. Jedes Öffnen eines Objektes (aber vor allem eines Werkzeugs oder Projekts) reduziert diesen freien Speicherplatz. Und genau für diesen Zweck ist die Speicheranzeige nämlich auch gedacht. Sie gibt Ihnen die Möglichkeit, recht genau den Speicherbedarf eines Programms zu ermitteln.

Wollen Sie häufig mit mehreren Werkzeugen gleichzeitig arbeiten, also das Multitasking so richtig nutzen, ist es sinnvoll, sich zu jedem Programm zu notieren, welchen Speicherbedarf es ungefähr hat. Zur Ermittlung dieses Wertes reicht meist ein Probelauf des Programms und die Kontrolle der Anzeige des freien Speichers in der Workbench-Titelleiste. Sie können dann jederzeit abschätzen, ob für ein weiteres Werkzeug, das sie gerade öffnen wollen, noch genügend Platz ist.

Ein weiteres Problem des Multitasking ist natürlich, daß sich jedes Programm, welches gleichzeitig mit anderen Programmen läuft, die Leistung des Computers mit diesen anderen Programmen teilen muß. Bei zwei oder drei Programmen, die gleichzeitig laufen, kann dies noch ganz erträglich sein, besonders dann, wenn diese Programme nicht sehr »rechenintensiv« sind, das heißt, den Computer nicht allzusehr beanspruchen. Je mehr Programme Sie dann aber darüber hinaus noch starten, desto langsamer erscheinen die Ergebnisse der Programme auf dem Bildschirm und desto langsamer werden auch die Reaktionen des Amiga 500 auf Ihre Eingaben.

Wenn Sie einen ungefähren Eindruck dieses Effekts haben wollen, brauchen Sie nur die Schublade *Demos* auf der Workbench-Diskette zu öffnen. (Falls Sie diese Schublade auf der Workbench-Diskette nicht finden, liegt Sie bei Ihnen auf der Extras-Diskette.) Darin befinden sich vier kleine Demonstrationsprogramme für die Fähigkeiten des Amiga – wie der Name *Demos* schon sagt. Sie eignen sich vor allem sehr gut als Demonstrationsobjekte für das Multitasking. Öffnen Sie dazu die vier Programme nacheinander. Verschieben und vergrößern Sie die dabei neu auftauchenden Fenster und betrachten Sie die grafischen Effekte ein wenig, die in den Fenstern zu sehen sind. Achten Sie dabei vor allem auf die Geschwindigkeit, mit der die Programme ablaufen. (Machen Sie das Fenster mit den Linien darin, *Lines*, recht groß, damit die Grafik gut sichtbar wird.)

Öffnen Sie nun die Programme nacheinander noch einmal, **ohne** vorher die schon erschienenen Fenster zu schließen. Achten Sie dabei darauf, wie die Zahl in der Titelleiste der Workbench immer kleiner wird und die Geschwindigkeit, mit der die Grafiken gezeichnet werden, bei jedem neuen Programm weiter abnimmt. (Wenn statt der Zahl in der Titelzeile der Name eines der Demoprogramme auftaucht, müssen Sie einmal in ein Disketten- oder Schubladenfenster klicken.) Auch die Reaktion der Workbench auf das Verschieben, Vergrößern und Verkleinern eines Fensters wird nun immer »träger«. Schließen Sie nun wieder langsam einige von den Fenstern der Demoprogramme und achten Sie darauf, wie die Geschwindigkeit in den übrigbleibenden Fenstern dabei wieder größer wird. (Das Schließen eines Fensters beendet bei jedem der Demoprogramme auch gleichzeitig das Programm.) Lassen Sie das Fenster mit den Linien darin bis zuletzt offen, dann erhalten Sie den spektakulärsten Effekt.

Sie haben nun live miterlebt, wie sich ein Computer aufteilt und dabei gehörig »in die Knie geht«. Denken Sie aber daran, daß nicht alle Programme immer so »beschäftigt« sind wie die Demoprogramme. Viele Werkzeuge auf der Workbench eignen sich sehr gut für ein zu anderen Programmen parallel laufendes Arbeiten und sie werden dabei nicht unerträglich langsamer.

5.1.3 Das Problem Bildschirm

Der Speicher ist aber nicht das einzige Problem, das das Multitasking mit sich bringt. Genauso wie den Speicher müssen sich mehrere parallel arbeitende Programme natürlich auch den Bildschirm (und die restlichen »Ressourcen«) des Amiga 500 teilen. Eine erste Lösung für dieses Problem stellen schon die Fenster dar. Mit der Verwendung eines Fensters hat ein Programm ja schon die Möglichkeit, recht unabhängig von dem zu werden, was die anderen Programme gerade tun. Das Programm kann zum Beispiel »so tun«, als hätte es den ganzen Bildschirm oder eine noch viel größere Fläche allein zur Verfügung und kann auf dieser ganzen Fläche Grafiken, Texte usw. zeigen. Sichtbar wird davon aber nur ein Teil, der so groß wie das Fensterinnere des Fensters ist, das dem Programm beim Start zugeteilt wurde. Der Programm-**Anwender**, nicht der Programmierer, entscheidet dann darüber, wie groß dieser Ausschnitt ist und welcher Teil des Gesamtbildes gezeigt wird.

Die Werkzeuge des Amiga, die Sie bisher kennengelernt haben, nutzen seine grafischen Fähigkeiten kaum. Sie öffnen alle nur ein Fenster auf der Workbench und begnügen sich mit den vier Farben, die die Workbench standardmäßig zur Verfügung stellt (Schwarz, Weiß, Blau und Orange, wenn Sie diese Farben noch nicht mit dem Programm *Preferences* geändert haben). Was aber geschieht, wenn einem Programm die Bildschirm-Attribute, wie sie die Workbench standardmäßig zur Verfügung stellt, »nicht gefallen«? Ein Programm kann zum Beispiel eine größere Anzahl oder eine andere Kombination von Farben bevorzugen (oder benötigen) als die Workbench. Der Amiga bietet ja noch wesentlich mehr Möglichkeiten als die vier Farben des Workbench-Bildschirms.

Ein solches Programm könnte nun natürlich einfach in den Interna des Amiga »herumpfuschen« und die gewünschten Einstellungen für die Bildschirm-Attribute vornehmen. Alle anderen Programme, die gerade aktiv sind, würden davon aber beeinträchtigt. Es könnte sein, daß deren Fenster auf einmal schwer oder gar nicht mehr lesbar wären. Es

könnte aber auch vorkommen, daß diese Programme gar nicht mehr lauffähig sind oder nur noch völliger Quatsch auf dem Bildschirm erscheint. Der Amiga 500 beziehungsweise die Amiga-Software bietet aber auch für diese Problematik eine Lösung, die sogenannten »Screens«.

5.2 Screens, die Superfenster

Glücklicherweise erlaubt es die Video-Hardware des Amiga 500, jedem Programm seinen eigenen »virtuellen« Bildschirm zur Verfügung zu stellen. (Statt virtuellen Bildschirm werde ich meist den Fachausdruck Screen verwenden; er ist einfach kürzer.) Diese Möglichkeit heißt deshalb virtueller Bildschirm, weil es natürlich nur einen wirklichen, physikalischen Bildschirm gibt, den Sie ja vor sich sehen.

Einem Programm kann es aber egal sein, ob es einen virtuellen Bildschirm verwendet oder einen echten; die Programmierung bleibt nahezu gleich. Nur Sie als Anwender müssen sich entscheiden, welchen virtuellen Bildschirm Sie auf Ihrem wirklichen Bildschirm sehen wollen. Wollen Sie (Teile von) mehrere(n) virtuellen Bildschirme(n) sehen, müssen Sie den wirklichen Bildschirm entsprechend »aufteilen«.

5.2.1 Wie Sie Screens bewegen

Vielleicht ist es Ihnen auf der Workbench schon öfter einmal passiert, daß Sie aus Versehen mit der linken Maustaste in die Titelleiste des Bildschirms geklickt haben, statt die rechte (Menü-)Taste zu verwenden. Wenn Sie bei einem solchen Fehler die Maus bewegen, kann es passieren, daß sich das Bild zu bewegen beginnt. Klickt man nämlich mit der linken Maustaste in die Titelleiste eines (virtuellen) Bildschirms, so kann man diesen ergreifen – wie ein Piktogramm oder ein Fenster – und auf und ab bewegen. Liegt »hinter« dem Workbench-Bildschirm noch ein anderer Bildschirm, so kommt dieser zum Vorschein, wenn Sie den Workbench-Bildschirm nach unten ziehen.

Probieren Sie es aus: Sie benötigen dazu allerdings die Extras-Diskette. Auf dieser befinden sich das Amiga-BASIC-Programm und einige Beispielprogramme in BASIC. Eines dieser Beispielprogramme müssen Sie nun starten. Legen Sie dazu die Extras-Diskette in das Diskettenlaufwerk (wenn Sie ein zweites Diskettenlaufwerk besitzen, können Sie die Diskette auch in dieses schieben). Öffnen Sie das Disketten-Piktogramm, sobald es auf der Workbench erscheint (unter dem Namen *ExtrasD*) und suchen Sie die Schublade *BasicDemos*. Öffnen Sie diese Schublade und suchen Sie das Piktogramm *Screen*. Dies ist ein BASIC-Programm, das Sie starten können, indem Sie es öffnen (zum Beispiel, indem Sie einen Doppelklick darauf machen). Starten Sie dieses Programm nun und warten Sie eine Weile, bis sich auf dem Bildschirm nichts mehr ändert. Der Bildschirm müßte dann etwa so aussehen wie das folgende Bild. Das es sich um einen neuen, vom Workbench-Screen verschiedenen Screen handelt, können Sie ganz leicht daran feststellen, daß in ihm wesentlich mehr als die vier auf der Workbench üblichen Farben vorkommen.

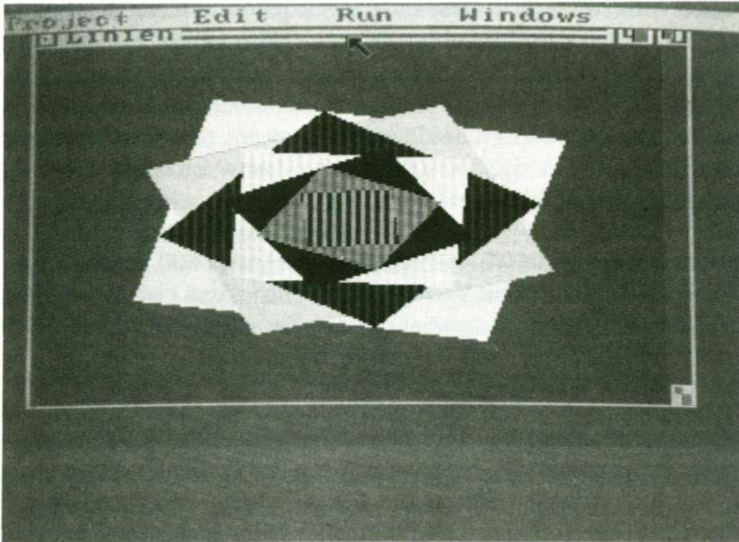


Bild 5.1: Das Programm Screen

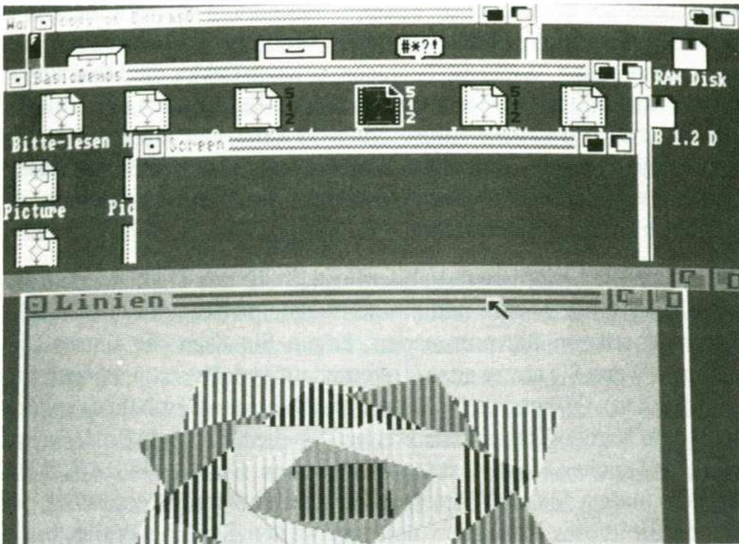


Bild 5.2: Zwei Screens gleichzeitig am Bildschirm

Der Umgang mit einem solchen Screen ist ganz einfach. Ein Screen verhält sich fast genauso wie ein Fenster. Sie können ihn an seiner Titelleiste ergreifen und auf- und abbewegen – jedoch nicht nach links oder rechts. Probieren Sie es aus: Gehen Sie mit dem Mauszeiger in die Titelleiste und halten Sie die Selektionstaste (die linke Maustaste) fest. Wenn Sie nun den Mauszeiger auf- und abbewegen, folgt das gesamte sichtbare Bild (mit einer gewissen »Trägheit«) dieser Bewegung und hinter diesem Bild wird der blaue Workbench-Bildschirm sichtbar.

5.2.2 Wie Sie den »Screen-Stapel« umschichten

Die Reihenfolge, in der Screens »im Stapel übereinanderliegen«, können Sie mit den Back- und Front-Gadgets am rechten Rand der Titelleiste eines Screens ändern. Sie funktionieren wie die gleich aussehenden Gadgets eines Fensters. Schieben Sie, um das auszuprobieren, den obenliegenden vielfarbigen Screen an den oberen Bildschirmrand und klicken Sie in sein Back-Gadget (eines der beiden kleinen Symbole am rechten Rand der Titelleiste). Sofort verschwindet dieser und der Workbench-Screen kommt nach oben. Dessen Titelleiste wird allerdings von einem Fenster namens *Screen* verdeckt. Machen Sie dieses Fenster nun möglichst klein und legen Sie es in die Bildschirmmitte. Darum herum müßten nun die Piktogramme und Fenster der Workbench auftauchen. Nun können Sie auch den Workbench-Screen an seiner Titelleiste mit der Maus ergreifen und auf- und abbewegen. Wenn Sie ihn nach unten schieben, müßte darunter wieder der mehrfarbige Screen des eben gestarteten BASIC-Programms zum Vorschein kommen.

Klicken Sie nun in das Back-Gadget des Workbench-Screens und dieser »fällt« wieder nach hinten. Probieren Sie diese beiden Gadgets und das Verschieben der Screens ruhig ein wenig aus, um ein Gefühl dafür zu bekommen. Sie können fast nichts dabei falsch machen. Und wenn Sie einmal nicht mehr weiter wissen sollten oder die Workbench nicht wiederfinden, starten Sie Ihren Amiga 500 einfach neu (mit [Ctrl]+[*C*=]+[A]). Meistens ist eine solch radikale Maßnahme aber überhaupt nicht nötig. Wenn Sie den Workbench-Screen zum Beispiel nicht wiederfinden können, so liegt das wahrscheinlich daran, daß Sie ihn bis ganz an den unteren Bildschirmrand gezogen haben. Legen Sie in diesem Fall einfach den anderen Screen nach hinten (mit dem Back-Gadget), gehen Sie mit der Maus ganz nach unten, drücken Sie die Selektionstaste und fahren Sie bei gedrückter Taste langsam nach oben.

In diesem Zusammenhang könnten vielleicht noch zwei weitere Tastenkombinationen wichtig für Sie sein. Diese Tastenkombinationen holen den Workbench-Screen ganz nach vorn oder legen ihn ganz nach unten hinter alle anderen Screens (es können auch mehr als nur ein anderer Screen sein). Diese Tastenkombinationen sind immer dann sehr praktisch, wenn Sie die Titelleiste des Workbench-Screens nicht finden oder wenn ein schlecht programmiertes Programm »seinen eigenen« Screen nach oben legt und dieser keine Titelleiste und kein Back-Gadget hat. Halten Sie in so einem Fall einfach die Commodore-Taste (links neben der Leertaste) fest und drücken Sie auf [N]. Der Workbench-Screen kommt dann nach vorn. Genau so können Sie den Workbench-Screen auch wieder nach hinten legen, indem Sie die Commodore-Taste festhalten und dann auf [M] drücken.

Probieren Sie diese beiden Tastenkombinationen auch am besten gleich aus – und prägen sie sich ein. (Zur Wiederholung deshalb noch einmal: [*C*=]+[N] holt den Workbench-Screen nach vorn und [*C*=]+[M] legt ihn nach hinten.) Besonders beim Herumprobieren mit neuen unbekannten Programmen sind diese »Tastentricks« oft die Rettung in letzter Not.

Wenn Sie zum Abschluß Ihrer Experimente das BASIC-Programm wieder verlassen wollen, klicken Sie einfach einmal in das Fenster *Linien* (mit der bewegten Grafik darin). Gehen Sie dann mit dem Mauszeiger in die Titelleiste, drücken Sie die Menütaste und wählen Sie zunächst den Befehl *Stop* aus dem Menü *Run.* Daraufhin verschwindet der bunte Screen. Klicken Sie nun in das Fenster *Screen* auf dem Workbench-Screen, worauf in diesem Fenster das Wort »Ok« auftaucht, und wählen Sie dann den Befehl *Quit* aus dem Menü *Project.* Falls Sie dieses Menü oder das Fenster *Screen* nicht finden sollten, oder wenn Sie es besonders eilig haben, tut es ein Neustart mit [Ctrl]+[*C*=]+[A] natürlich auch. Mehr zum Thema BASIC können Sie in den letzten Kapiteln dieses Buches erfahren.

Nachdem Sie nun auch die Screens kennengelernt haben, kennen Sie jetzt alle wichtigen »Bausteine« der Amiga-Software. Damit ist der kleine »Amiga-Grundkurs«, den die ersten 5 Kapitel dieses Buches darstellen, fast abgeschlossen. In den folgenden Abschnitten dieses Kapitels werden Sie nun noch die restlichen Werkzeuge (Programme) kennenlernen, die Sie beim Kauf des Amiga 500 dazu erhalten. Den Anfang macht *Preferences.*

5.3 Preferences – das wichtigste Werkzeug überhaupt

Vielleicht das wichtigste Werkzeug der Workbench überhaupt ist *Preferences* (zu deutsch etwa *Vorlieben;* besser könnte man den Namen aber vielleicht mit *Einstellungen* oder *Grundeinstellungen* übersetzen). Mit diesem Werkzeug können Sie zum Beispiel Uhrzeit und Datum stellen, die Farben ändern, in denen Ihnen die Workbench am Bildschirm gezeigt wird, und dem Amiga 500 mitteilen, welchen Drucker Sie an welchem Stecker angeschlossen haben (das muß man ihm nämlich erst extra sagen; er merkt es nicht von selbst).

5.3.1 Das Hauptfenster von Preferences

Sie finden *Preferences* im Diskettenfenster der Original-Workbench-Diskette und können das Programm wie jedes andere auch mit einem Doppelklick starten. Danach erscheint ein bildschirmfüllendes Fenster, das geradezu überläuft von einer Ansammlung der verschiedensten Gadgets. Dies ist das »Hauptfenster« von *Preferences.* Beim Klick auf einige der verschiedenen Knöpfe in diesem Fenster erscheinen noch bis zu vier andere Fenster, die aber nie gleichzeitig mit dem Hauptfenster zu sehen sind, sondern es immer (vorübergehend) verdecken. Nachdem Sie Preferences aufgerufen haben, können Sie es nur von diesem Fenster aus wieder verlassen. Unten rechts im Fenster sind fünf Knöpfe zu sehen: *Change Printer, Edit Pointer, Save, Use* und *Cancel.* Wenn Sie drei davon (mit einem Mausklick) betätigen, so wird *Preferences* verlassen, die beiden anderen Knöpfe zeigen Ihnen zwei der anderen Fenster von Preferences.

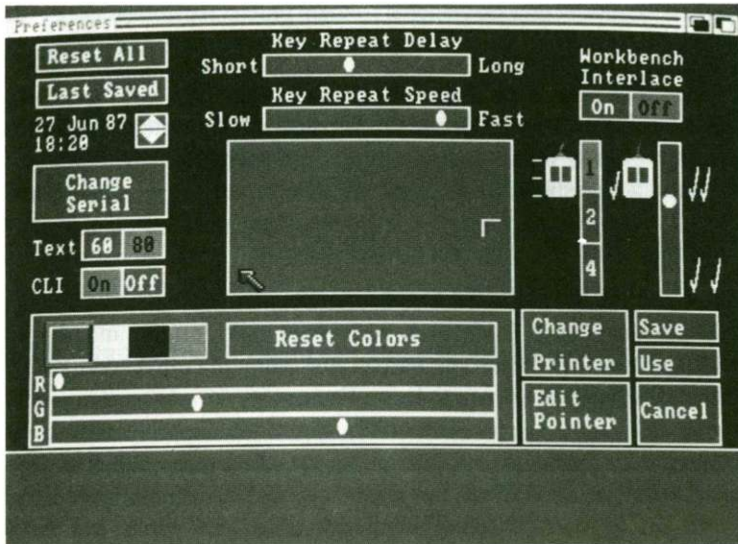


Bild 5.3: Das Hauptfenster von Preferences

5.3.2 Wichtige Tasten im Preferences-Hauptfenster

Drücken Sie auf *Cancel* (engl. für *Abbrechen* oder *Vergiß es*), so wird *Preferences* verlassen und alle Änderungen, die Sie eventuell zuvor durchgeführt haben, werden ignoriert. Drücken Sie auf *Save*, so werden die Änderungen erstens gültig und zweitens auf Ihrer Workbench-Diskette gespeichert. Sie sind dann auch nach dem nächsten Neustart beziehungsweise Einschalten noch gültig. Betätigen Sie hingegen *Use* (engl. *to use = verwenden/benutzen*), so werden Ihre Änderungen und Einstellungen zwar gültig, sie werden aber nicht auf der Diskette gespeichert. Das heißt, nach dem nächsten Neustart oder Aus- und Einschalten des Amiga 500 gelten wieder die alten Einstellungen, die Sie beim Start von Preferences vorgefunden haben.

Save funktioniert natürlich nur, wenn der Schreibschutz Ihrer Workbench-Diskette (der kleine Plastikriegel in einer Ecke der Diskette) nicht aktiviert ist. Ist das der Fall, erscheint eine Fehlermeldung. Sie können dann die Diskette herausnehmen, den Schreibschutz entfernen (den Plastikriegel verschieben) und die Diskette wieder einlegen. Haben Sie es sich anders überlegt und wollen Ihre Änderungen lieber doch nicht auf der Diskette abspeichern, klicken Sie einfach auf den Knopf *Cancel* in dem Fenster, das Ihnen die Fehlermeldung zeigt.

In der oberen linken Ecke des Hauptfensters befinden sich zwei weitere wichtige Knöpfe: *Reset All* und *Last Saved*. Wenn Sie *Reset All* (auf deutsch etwa *alles zurücksetzen*) anklicken, gelten für alle Einstellungen (Bildschirmfarben und so weiter), die Sie mit Preferences verändern können, Standardwerte, die in der Systemsoftware des Amiga 500 festgelegt sind. Dies sind die Einstellungen, mit denen Sie wahrscheinlich beim ersten Einschalten Ihres Amiga gearbeitet haben. Klicken Sie auf *Last Saved* (auf deutsch etwa *zuletzt gesichert*), so gelten die

Einstellungen, die Sie das letzte Mal gesichert haben, indem Sie Preferences mit einem Klick auf *Save* verlassen haben. Wenn Sie noch nie mit Preferences gearbeitet haben, ergeben *Reset All* und *Last Saved* dasselbe Ergebnis – eben die Standardeinstellungen.

Der Schalter *CLI* (rechts im unteren Drittel des Fensters) wird in einem späteren Kapitel noch näher erläutert. Er bestimmt, ob das sogenannte CLI auf der Workbench sichtbar ist oder nicht und wird hier nur der Vollständigkeit halber erwähnt.

5.3.3 Ganz nach Geschmack: die Workbench-Farben

Zu den Einstellungen, die Sie in *Preferences* ändern können, gehören auch die vier Farben, die der Amiga 500 im Workbench-Bildschirm verwendet. Alle Programme, die keine eigenen Screens erzeugen, müssen mit diesen vier Farben auskommen. Zum Ändern dieser vier Farben dienen die drei »Schieberegler« unten links im Fenster.

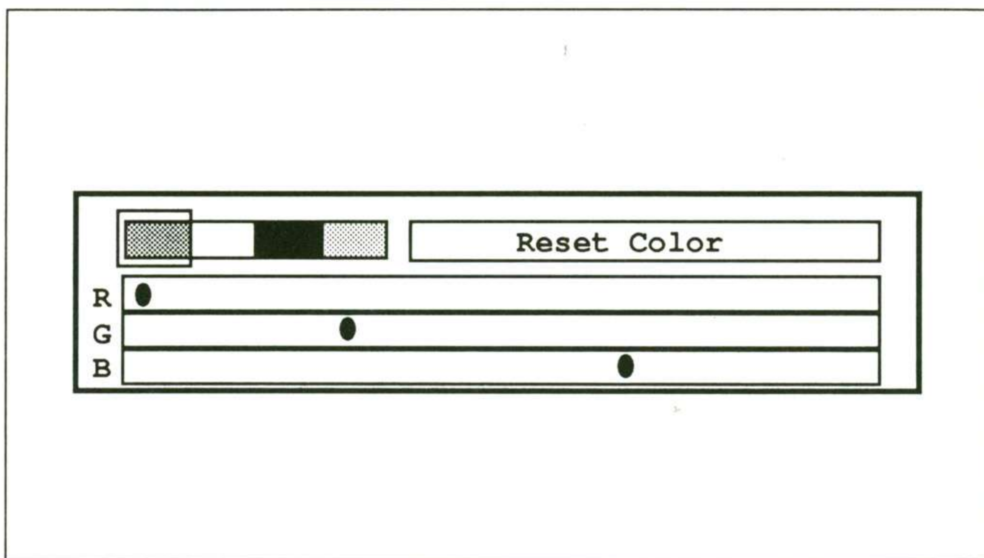


Bild 5.4: *Einstellen der Workbench-Farben*

Wollen Sie die Farben ändern, müssen Sie zunächst eine aussuchen. Dies geschieht, indem Sie eines der vier farbigen Quadrate, die über den Reglern zu sehen sind, anklicken. Die so selektierte Farbe wird dann mit einem Rahmen versehen. Die kleinen Knöpfe in den Reglern darunter können Sie dann mit der Maus ergreifen und nach links und rechts verschieben, wobei sich die zuvor ausgewählte Farbe sofort ändert. Die Buchstaben R, G und B neben den Reglern stehen für Rot, Grün und Blau. Aus diesen drei Grundfarben werden die am Bildschirm sichtbaren Farben nämlich »zusammengemischt«. Sind alle Regler am rechten Anschlag, ergibt dies die Farbe Weiß. Sind alle am linken Anschlag, entsteht Schwarz. (Auf dieselbe

Weise – durch additive Farbmischung aus Rot, Grün und Blau – entstehen übrigens auch die Farben auf Ihrem Farbfernseher.)

Es ist nicht ganz einfach, eine Farbe, die man sich vorstellt, auf diese Weise zusammenzumischen. Ein wenig Experimentieren wird Ihnen aber bestimmt bald ein Gefühl dafür vermitteln. Sie sollten ruhig ein wenig mit den drei Reglern herumspielen! Ähnliche Regler werden in fast jedem Mal- und Grafik-Programm für den Amiga verwendet. Falls es Ihnen gelungen sein sollte, so scheußliche Farben zusammenzumischen, daß Ihre Augen es nicht mehr aushalten oder Ihnen übel wird, können Sie übrigens mit dem Knopf *Reset Colors* die Standard-Kombination (Schwarz, Weiß, Orange und Blau) wiederherstellen.

5.3.4 Wem die Stunde schlägt: Einstellen von Uhrzeit und Datum

Während die Verstellung der Workbench-Farben eher in die Kategorie Spielerei fällt, ist die folgende Funktion von Preferences durchaus ernsthaft. Oben rechts im Fenster sehen Sie die aktuelle Uhrzeit und das Datum – oder besser, was der Amiga dafür hält. Wenn Sie Wert darauf legen, daß seine Vorstellung von Zeit und Datum mit der Ihren übereinstimmt, so können Sie die Anzeige mit den beiden Pfeilen daneben korrigieren.

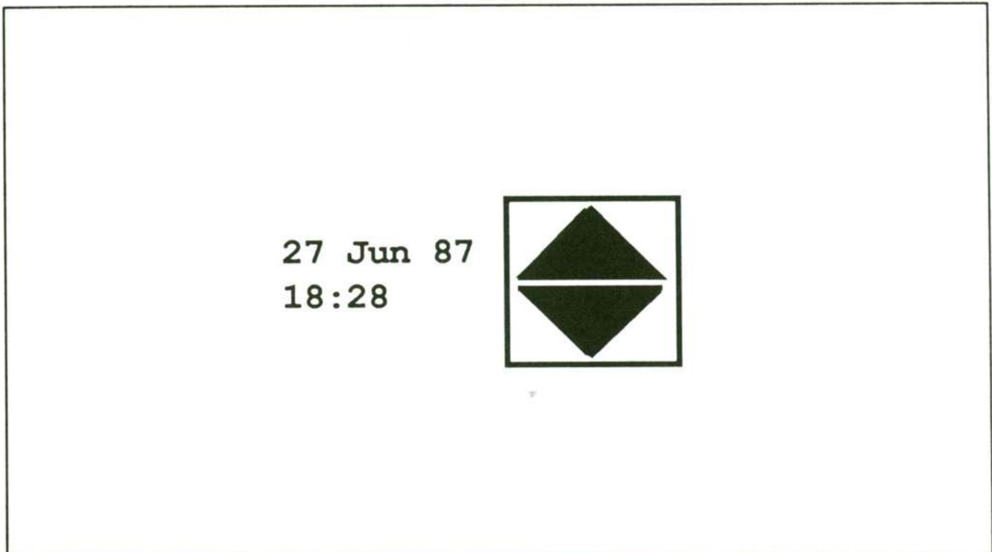


Bild 5.5: *Einstellen von Uhrzeit und Datum*

Klicken Sie dazu erst auf die Ziffer (beziehungsweise die Monatsbezeichnung), die Sie verstellen wollen. Danach können Sie diese mit einem Klick auf die danebenstehenden Pfeile verändern. Der Pfeil nach oben erhöht die zuvor selektierte Ziffer um eins, der Pfeil nach unten macht sie entsprechend kleiner. So können Sie nacheinander alle angezeigten Zahlen auf die

korrekten Werte stellen. Wenn Sie damit fertig sind, geht die Uhr Ihres Amiga 500 richtig. Solange Sie ihn eingeschaltet lassen und keinen Neustart machen, behält sie auch die richtige Zeit bei. Wenn Sie ihn jedoch abschalten, vergißt er die richtige Uhrzeit – sofern Sie nicht die Speichererweiterung Amiga 501 mit integrierter Uhr eingebaut haben. Mit dieser Uhr vergißt Ihr Amiga 500 Uhrzeit und Datum nicht mehr, wenn Sie sie einmal richtig eingestellt haben.

5.3.5 Das Kleingedruckte: Einstellen von Schriftgröße und Interlace

Die nächste Einstellungsmöglichkeit betrifft die Größe (und damit auch Lesbarkeit) von Text auf dem Bildschirm. Sie können zwischen zwei Größen wählen: 60 Zeichen pro Zeile und 80 Zeichen pro Zeile. (Text mit 80 Zeichen pro Zeile wird auf einfacheren Bildschirmen oder gar auf einem Fernseher leicht unleserlich, da es dem Bildschirm an Schärfe mangelt. 60spaltiger Text kann hingegen fast immer scharf dargestellt werden.) Je nachdem, welchen Bildschirm Sie an Ihren Amiga angeschlossen haben, wie nahe Sie dem Bildschirm sind und wie gut Ihre Augen sind, wird eine dieser beiden Einstellungen ideal für Sie sein. Klicken Sie einfach auf eine der beiden Zahlen neben dem Wort *Text*. Die so ausgewählte Zahl erscheint (sofern Sie nicht an den Farbreglern gespielt haben) schwarz auf orange, während die andere weiß auf blau zu lesen ist.

Wenn Sie einen sehr guten Bildschirm besitzen (und recht widerstandsfähig gegenüber Kopfschmerzen sind), können Sie sogar noch mehr Informationen auf den Bildschirm packen als mit dem 80-Zeichen-Modus allein. Oben rechts im Hauptbildschirm von Preferences befinden sich unter der Überschrift *Workbench Interlace* zwei Knöpfe mit den Beschriftungen *On* und *Off* (Ein und Aus). Wenn Sie den Knopf *On* anklicken, *Preferences* dann mit *Save* verlassen und einen Neustart machen, wird die Workbench im *Interlace*-Modus gezeigt. Dabei passen doppelt soviele Zeilen auf den Bildschirm als im »normalen« Bildschirmmodus. Dafür flimmert das Bild aber auch verhältnismäßig stark. Dieses Flimmern können Sie aber durch geschickte Wahl der Workbench-Farben reduzieren. (Mehr zum Interlace-Modus, die Grafik-Hardware des Amiga und die Gründe für das starke Flimmern erfahren Sie im letzten Teil dieses Buches.)

Beachten Sie bitte, daß ein Wechsel zwischen Interlace-Modus und dem normalen Bildschirm-Modus nur nach einem Neustart und vorherigem Verlassen von *Preferences* durch Anklicken von *Save* stattfindet! Alle anderen in *Preferences* durchgeführten Änderungen werden dagegen sofort nach Anklicken von *Use* oder *Save* gültig.

5.3.6 Nichts für Schwindelige: Zentrieren des Bildes

Einen recht spektakulären Effekt können Sie mit dem großen blauen Rechteck in der Mitte des Hauptfensters von *Preferences* erzielen. Dieses Rechteck dient zur Zentrierung des Bildes auf dem Bildschirm. Durch kleine Unterschiede bei den Bauteilen des Bildschirms kann es nämlich dazu kommen, daß das Bild, das der Amiga 500 erzeugt, nicht immer genau in der Mitte der Fläche zu sehen ist. Manchmal sind sogar Teile des Bildes durch den Rand verdeckt.

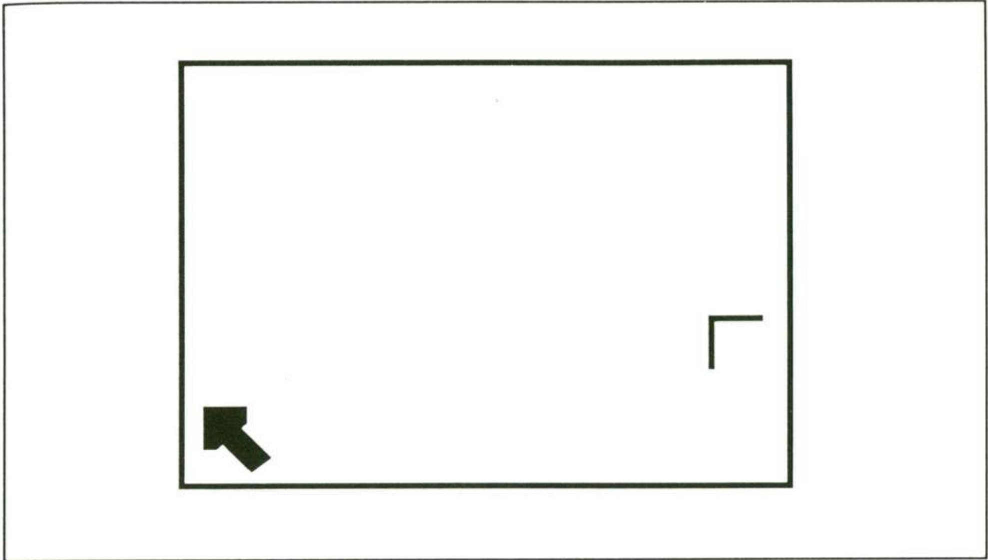


Bild 5.6: *Zentrierung des Bildes mit der Maus*

»Ergreifen« Sie einfach den kleinen weißen Winkel in der blauen Fläche mit der Maus und bewegen Sie diese bei gedrückter Selektionstaste. Dann folgt nicht nur dieser Winkel den Maus-bewegungen, sondern das komplette Bild. Auf diese Weise ist es problemlos möglich, das Bild innerhalb des Bildschirms dahin zu legen, wo man es haben möchte.

5.3.7 Einstellen der automatischen Tastenwiederholung

Mit den beiden Schiebereglern am oberen Rand des Hauptfensters können Sie die Geschwindigkeit der automatischen Tastenwiederholung und die Pause bis zu deren Beginn einstellen. Halten Sie nämlich eine der Tasten fest, während Sie einen Text eingeben oder bearbeiten, so wirkt dies nach einer kleinen Pause so, als würden Sie dieselbe Taste schnell hintereinander immer wieder niederdrücken; die Taste wiederholt sich automatisch.

Die Geschwindigkeit, mit der die simulierten Tastendrücke ablaufen, können Sie mit dem Regler unter der Überschrift *Key Repeat Speed* einstellen. Je näher der Knopf in diesem Regler dem rechten Anschlag (engl. *Fast = schnell*) kommt, desto höher ist die Wiederholungsrate. Der Regler *Key Repeat Delay* bestimmt darüber, wie lange Sie eine Taste niederhalten müssen, bis die automatische Wiederholung beginnt. Je näher der Knopf in diesem Regler dem rechten Anschlag (engl. *Long = lange*) kommt, desto länger ist diese Pause.

Wie Sie diese beiden Regler einstellen, ist weitgehend persönlicher Geschmack. Falls Sie aber kein »Zehn-Finger-Tipper« sind, sollten Sie den Regler *Key Repeat Delay* nicht auf zu kleine Werte (nahe bei *Short = kurz*) einstellen, da sonst fast jede Taste gleich zwei Buchstaben

erzeugt. Auch zu hohe Einstellungen für *Key Repeat Speed* führen in vielen Programmen zu verwirrenden Effekten und sind deshalb nicht zu empfehlen.

5.3.8 Einstellen der »Mausübersetzung«

Zwei andere wichtige Einstellungen ähnlicher Art sind die »Übersetzung« der Mausbewegungen und die Zeitspanne, während dieser zwei Mausklicks als ein Doppelklick gezählt werden. Diese beiden Werte können Sie mit zwei Reglern am rechten Rand des Hauptfensters von *Preferences* einstellen.

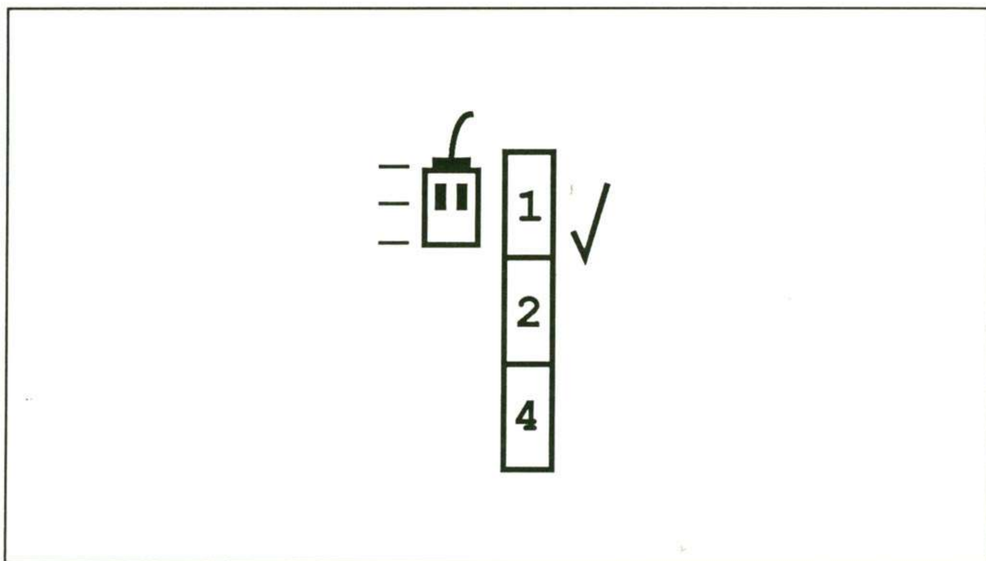


Bild 5.7: Einstellung der Mausübersetzung

Die Übersetzung der elektrischen Impulse, die von der Maus kommen, in die Bewegung des Mauszeigers am Bildschirm geschieht ja in der Systemsoftware des Amiga. Diese Umsetzung kann deshalb auch in der Software (durch *Preferences*) beeinflußt werden. Je höher der Wert ist, den Sie mit dem oberen der beiden Regler am rechten Fensterrand einstellen, desto weiter müssen Sie die Maus auf dem Tisch bewegen, um mit dem Mauszeiger eine bestimmte Strecke auf dem Bildschirm zurückzulegen.

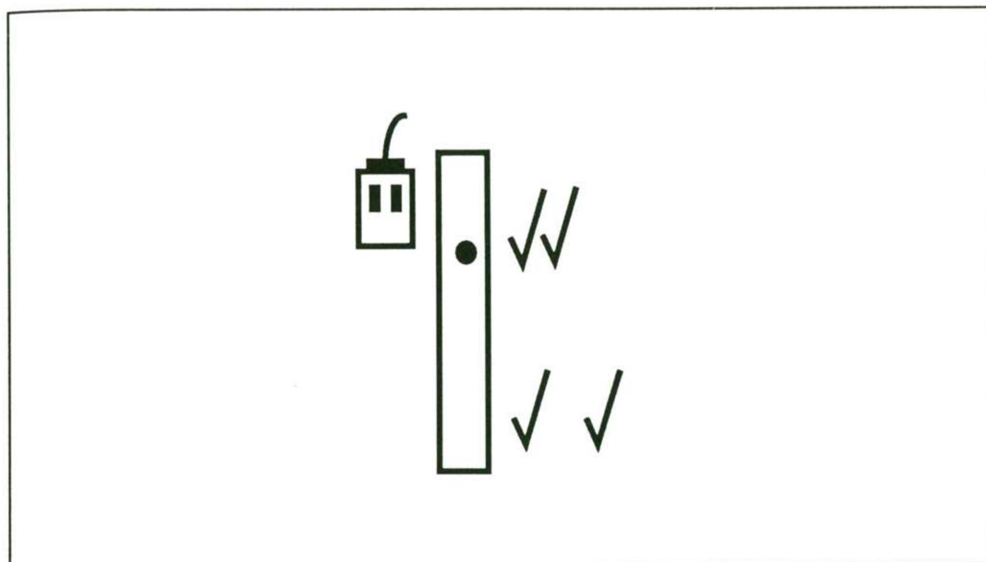


Bild 5.8: Einstellung der Doppelklick-Zeitspanne

Für viele Operationen auf der Workbench (besonders für das Öffnen von Objekten) ist es sehr wichtig, ob zwei Mausklicks, die kurz hintereinander erfolgen, wirklich zwei Mausklicks oder ein »Doppelklick« sind. Ist die Zeitspanne, während der die Amiga-Software noch zwei Klicks zu einem Doppelklick zusammenfaßt, zu groß, kann es sehr leicht vorkommen, daß Objekte versehentlich geöffnet werden – was Zeit kostet. Ist diese Zeitspanne zu kurz, kann es besonders für Anfänger sehr schwierig werden, überhaupt einen Doppelklick durchzuführen. Mit dem entsprechenden Regler im Hauptfenster von *Preferences* kann sich jeder Anwender diese Zeitspanne so einstellen, wie er es am angenehmsten empfindet. Je näher der Knopf in diesem Regler dem oberen Anschlag kommt, desto kürzer wird der bewußte Zeitraum. (Eine solch kurze Zeitspanne ist nur für geübte Benutzer geeignet.)

5.3.9 Ändern des Mauszeigers

Klickt man im Hauptfenster von *Preferences* auf den Knopf mit der Bezeichnung *Edit Pointer* (nahe der unteren rechten Ecke des Fensters), so erscheint ein neues Fenster, in dem man die Form und die Farben des normalerweise pfeilförmigen Mauszeigers ändern kann.

Auch für dieses Fenster gilt, daß, wenn man es durch Anklicken von *Cancel* (= Abbrechen) abschließt, alle Modifikationen am Mauszeiger ignoriert werden. Verläßt man es aber mit einem Klick auf OK, so kommt man mit einem neuen Mauszeiger ins Hauptfenster zurück.

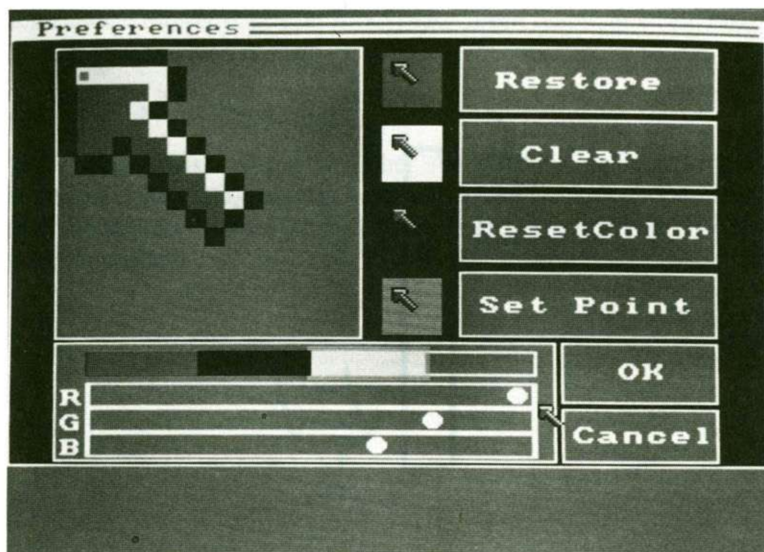


Bild 5.9: Ändern des Mauszeigers

Die Änderung der Mauszeiger -Farben erfolgt in ähnlicher Weise wie die Änderung der Farben für die Workbench (siehe oben). Drei Regler am unteren Rand des Fensters sind hierfür »zuständig«. Bei einem Mauszeiger können Sie aber nur drei Farben modifizieren, die vierte ist »transparent«. Das heißt, die darunterliegende Farbe der Workbench »scheint durch«.

In einem großen Rechteck oben links im Fenster können Sie den Mauszeiger Punkt für Punkt ändern. Er wird dazu in circa achtfacher Vergrößerung gezeigt. Wollen Sie einem Punkt im Mauszeiger eine andere Farbe geben, so klicken Sie diese neue Farbe in der Palette über den drei Schiebereglern einfach an und klicken danach den entsprechenden Punkt in der vergrößerten Darstellung an. Wollen Sie einen Punkt löschen, müssen Sie einfach auf die »transparente« Farbe in der Palette klicken (normalerweise ist sie blau) und dann den zu löschenden Punkt anklicken. Wie dieser neue Mauszeiger aussieht, wenn er über den vier möglichen Farben der Workbench erscheint, können Sie rechts neben dem Vergrößerungsfeld gleich kontrollieren.

Zum Schluß müssen Sie noch den »heißen Punkt« des Zeigers bestimmen. Jeder Mauszeiger besitzt auch einen solchen »heißen Punkt«. Dies ist der Punkt, mit dem Sie »wirklich« zeigen, wenn Sie die Maustaste niederdrücken. Der ganze Zeiger ist ja für präzise Positionsbestimmungen zu vage. Beim normalen pfeilförmigen Zeiger ist die Spitze der heiße Punkt. Auch diesen Punkt können Sie verschieben. Betätigen Sie dazu den Knopf *Set Point* (engl. für *Punkt setzen*) und klicken Sie dann in der Vergrößerung den Punkt an, der der heiße Punkt werden soll. Nun brauchen Sie dieses Fenster nur noch mit einem Klick auf OK verlassen und haben schon einen neuen Mauszeiger.

5.3.10 Einstellen der seriellen Schnittstelle

Klickt man im Hauptfenster von *Preferences* auf den Knopf mit der Bezeichnung *Change Serial* (nahe der oberen linken Ecke), so erscheint ein neues Fenster, in dem man die Parameter der seriellen Schnittstelle ändern kann. Sie sollten das immer tun, wenn Sie ein neues Gerät an die serielle Schnittstelle des Amiga 500 angeschlossen haben (zum Beispiel einen Drucker oder ein Modem). Legen Sie dazu am besten das Handbuch dieses Geräts neben den Amiga. Die Einstellung einer seriellen Schnittstelle ist eine Kunst, da es dabei sehr viele Möglichkeiten gibt.

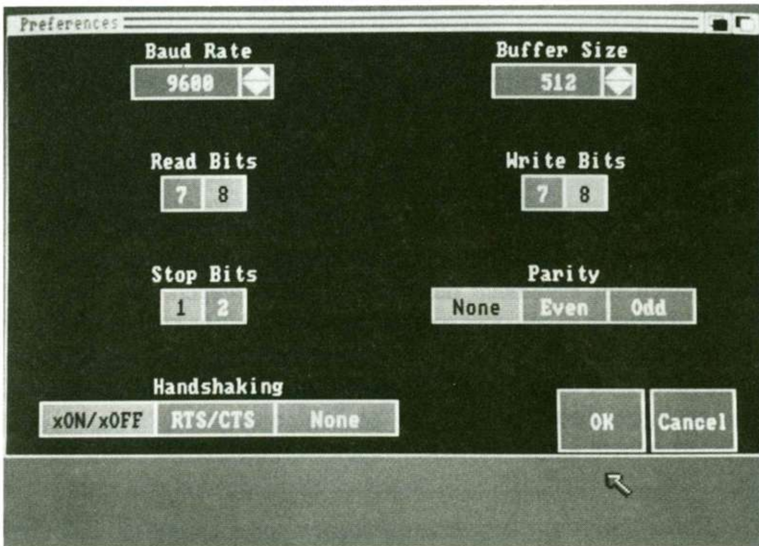


Bild 5.10: *Einstellen der seriellen Schnittstelle*

Alle Einstellungen in diesem Fenster können Sie mit wenigen Mausklicks erledigen. Entweder sind es Zahlen, dann können Sie sie mit einem Klick auf einen der beiden Pfeile neben der jeweiligen Zahl erniedrigen oder erhöhen. Oder es handelt sich um die Wahl zwischen mehreren sich ausschließenden Möglichkeiten, dann genügt der Klick auf die gewünschte Möglichkeit.

Am wichtigsten ist die Einstellung *Baud Rate* (Baudrate) oben links in diesem Fenster. Die Baudrate bestimmt die Geschwindigkeit der Datenübertragung durch eine serielle Schnittstelle. Übliche Werte für den Anschluß eines Modems sind 300 und 1200 Baud. Sehr schnelle Modems (die die Bundespost bislang noch nicht zuläßt) schaffen auch 2400 und 4800 Baud. Serielle Drucker erhalten ihre Daten (Buchstaben) hingegen meist mit 1200 oder 9600 Baud. Wenn Sie einen anderen Computer direkt über ein kurzes Kabel an den Amiga

anschließen, werden Sie meist mit 9600 oder 19.200 Baud arbeiten. Dabei können aber schon recht häufig Übertragungsfehler auftauchen.

Übertragungsfehler kann man unter Umständen durch eine Erhöhung der Puffergröße (*Buffer Size* rechts oben im Fenster) vermeiden. Ein größerer Datenpuffer gestaltet die Datenübertragung oft schneller und fehlerfreier. Besonders, wenn Sie mit einer hohen Baudrate arbeiten und bei *Handshaking* die Option *None* gewählt haben (siehe unten), kann der Puffer gar nicht groß genug sein.

Read Bits und *Write Bits* in der Mitte des Fensters bestimmen, wieviele Bits eines Bytes über die serielle Schnittstelle gesendet (*Write Bits*) und empfangen (*Read Bits*) werden. Da ein Byte aus acht Bits besteht, sollten Sie es bei beiden Einstellungen zunächst einmal mit der Option *8* versuchen. Besonders bei der Datenfernübertragung kann es jedoch Fälle geben, bei denen nur jeweils 7 Bit übertragen werden dürfen. Dies betrifft aber nur den Anschluß eines Modems; moderne Drucker benötigen nahezu ausschließlich 8 Bits. Von wenigen, sehr seltenen Ausnahmen abgesehen, sollten Sie bei *Read Bits* und *Write Bits* immer dieselbe Einstellung wählen.

Zu den Schaltern *Parity*, *Stop Bits* und *Handshaking* kann man sehr wenig sagen, ohne ausführlich zu beschreiben, wie serielle Datenübertragung vonstatten geht. Sie sollten die nötigen Einstellungen im Handbuch des Gerätes nachschlagen, das Sie an die serielle Schnittstelle angeschlossen haben oder so lange probieren, bis es klappt. Weit verbreitet sind heutzutage die Einstellungen *1* bei *Stop Bits* und *None* bei *Parity*. Bei *Handshaking* sollten Sie immer zunächst XON/XOFF oder RTS/CTS versuchen, bevor Sie *None* wählen. Die Option *None* erhöht gerade bei hohen Baudraten die Fehlerwahrscheinlichkeit bei der Datenübertragung, die sich nicht immer durch Erhöhung der Puffergröße (siehe oben) verringern läßt.

Wenn Sie diese Einstellungen alle vorgenommen haben – oder zumindest einen ersten Versuch gemacht haben – sollten Sie dieses Fenster mit einem Klick auf OK verlassen. Ein Klick auf CANCEL bringt Sie zwar auch ins Hauptfenster zurück, bewirkt aber – wie üblich – ansonsten nichts.

5.3.11 Die Druckeranpassung

Preferences dient auch dazu, Ihren Amiga an die verschiedensten Drucker anzupassen, die Sie an ihn anschließen können. Nachdem Sie den Drucker mit einem passenden Kabel hinten an Ihren Amiga angeschlossen haben, betätigen Sie dazu im Hauptfenster von *Preferences* die Taste *Change Printer* (engl. für *Drucker wechseln* beziehungsweise *Drucker anpassen*).

Auch für das Druckeranpassungsfenster, das daraufhin auftaucht, gilt, daß man es mit einem Klick auf CANCEL (abbrechen) oder mit OK wieder schließen und die Anpassung damit beenden kann. CANCEL bedeutet dabei Ignorieren aller Änderungen und OK Bestätigen der durchgeführten Anpassung, die dadurch gültig wird.

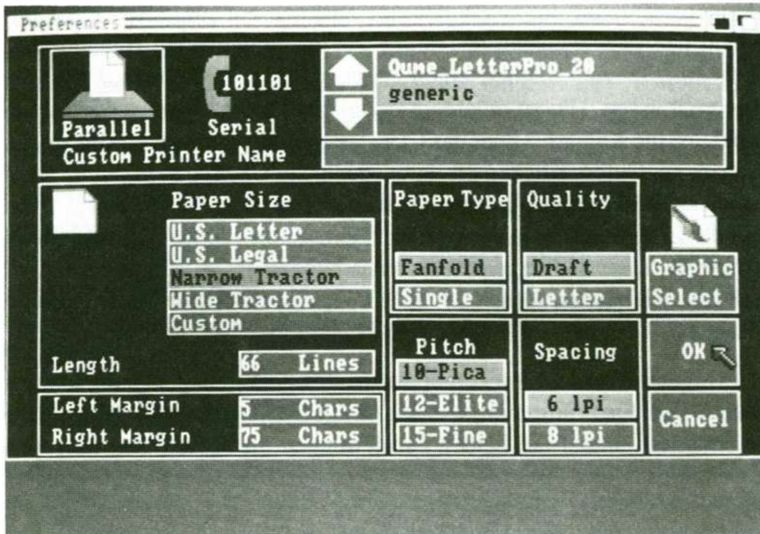


Bild 5.11: Auswahl des angeschlossenen Druckers

Welche Einstellungen Sie hier vornehmen müssen, hängt in starkem Maße von dem Drucker ab, den Sie verwenden. Konsultieren Sie deshalb das entsprechende Handbuch, das dem Drucker beilag. Falls Ihnen die darin auftauchenden Fachausdrücke alle oder teilweise nichts sagen, können Sie die Druckeranpassung auch durch den Händler vornehmen lassen und ein für allemal auf Ihrer Workbench-Diskette abspeichern. Sie brauchen sie im allgemeinen nie wieder zu verändern.

Die wichtigsten Einstellungen befinden sich in diesem Fenster am oberen Rand. Links können Sie bestimmen, an welchen Anschluß an der Rückseite des Amiga 500 Sie das Druckerkabel gesteckt haben. Klicken Sie dazu auf das Symbol *Parallel* oder *Serial*. Die beiden Symbole ähneln denen, die unter den entsprechenden Anschlüssen aufgedruckt sind. Rechts daneben müssen Sie angeben, welchen Druckertyp Sie angeschlossen haben. Hierzu müssen Sie solange auf die beiden nach oben und nach unten weisenden Pfeile klicken, bis der Name Ihres Druckers daneben hervorgehoben wird (das ist immer der mittlere der drei Namen). Taucht der Name Ihres Druckers nicht in dieser Liste auf, haben Sie Pech. Schlagen Sie dann im Handbuch Ihres Druckers nach, ob dieser vielleicht zu einem anderen Drucker »kompatibel« ist. Versuchen Sie dann den Namen dieses kompatiblen Druckers zu finden und wählen Sie diesen aus. Falls Sie auch keinen kompatiblen Drucker finden können, müssen Sie *Custom* auswählen, dann das Text-Gadget *Custom Printer Name* darunter anklicken und dort »Generic« (ohne Anführungsstriche) eintippen. Wenn Sie Glück haben, können Sie dann Texte drucken, aber keine Bilder oder Grafiken.

Wenn Sie einen Drucker kaufen, der nicht in der Liste steht, die Sie mit den beiden Pfeilen durchblättern können, und dieser Drucker vom Hersteller speziell an den Amiga 500 angepaßt

worden ist, kann es sein, daß der Hersteller eine Diskette mit spezieller Software mitliefert. Folgen Sie dann den Anweisungen des dem Drucker beiliegenden Handbuchs, wobei Sie wahrscheinlich eine spezielle Datei, einen sogenannten »Treiber«, auf Ihre Workbench-Diskette kopieren müssen. Hierzu müssen Sie vielleicht das CLI verwenden. In diesem Fall sollten Sie zunächst die Kapitel 7 über das CLI lesen. Wenn Sie damit fertig sind, können Sie wieder *Preferences* aufrufen. Klicken Sie dann wieder auf *Change Printer* und wählen Sie den Namen der eben kopierten Datei aus der Liste rechts oben durch Anklicken aus. Taucht dieser Name nicht in der Liste auf, müssen Sie die Option *Custom* wählen. Klicken Sie in das Text-Gadget unterhalb der Liste und tragen Sie dort den Namen des »Treibers« ein, den Sie gerade kopiert haben. (Diesen Namen sollten Sie auch dem Handbuch entnehmen können, das dem Drucker beiliegt.)

Die Gadgets im unteren Teil des Druckeranpassungsfensters sind nicht ganz so wichtig wie die beiden im oberen Teil. Sie bestimmen, wie der Amiga 500 Text und Grafik ausdruckt. Hierzu müssen aber zunächst einmal die beiden Einstellungen im oberen Teil korrekt sein, sonst funktioniert gar nichts. Unter *Paper Size* können Sie die Größe des Papiers einstellen, die Sie in Ihrem Drucker verwenden. Hierzu gibt es wieder einige (in den USA) übliche Längen zur Auswahl. Das bei uns übliche Papierformat DINA4 ist leider nicht dabei. Zur Angabe solcher (in den USA) unüblichen Längen gibt es die Option *Custom*. Klicken Sie auf diesen Knopf und dann in das Text-Gadget darunter (links daneben steht *Length*). Tragen Sie dann in diesem Text-Gadget ein, wie lang das von Ihnen verwendete Papier ist. Für normales DIN-A4-Papier müssen Sie 69 eintragen und für das übliche perforierte Endlospapier 72. Beim Ausdruck längerer Texte merken Sie, ob diese Längenangabe stimmt oder nicht. Wenn Sie eine zu große Länge eintragen, beginnt der Ausdruck der zweiten Seite zu tief unten auf dem Papier. Bei einer zu geringen Längenangabe beginnt der Text der zweiten Seite zu früh – vielleicht noch auf der ersten Seite des Papiers. Rechts neben der Auswahl der Papiergröße sollten Sie noch auswählen, ob Sie einzelne Blätter (*Single*) oder Endlospapier (*Fanfold*) verwenden. Klicken Sie das entsprechende Gadget an.

Links unten im Druckeranpassungsfenster können Sie angeben, ob Sie beim Ausdruck links und rechts auf dem Papier unbedruckte Ränder (engl. *margins*) haben wollen. Tragen Sie neben *Left Margin* ein, wieviel Platz am linken Rand freibleiben soll, und neben *Right Margin*, wieviel rechten Rand die Ausdrucke haben sollen. Beide Angaben müssen in Buchstabenbreiten (*Chars*) gemacht werden, wobei Sie sich ungefähr daran orientieren können, daß auf die volle Papierbreite circa 80 Buchstaben passen.

Unter der Überschrift *Pitch* können Sie die Druckdichte bestimmen. Beim Ausdruck von Grafiken bestimmt dies die Breite und die Proportionen des Bildes, beim Ausdruck von Texten die Breite (und manchmal die Form) der Buchstaben. *10-Pica* ergibt die breitesten Buchstaben beziehungsweise Bilder und *15-Fine* die schmalsten.

Genauso können Sie unter *Spacing* die vertikale Dichte des Drucks bestimmen. Die Option *6 lpi* (*lines per inch = Zeilen pro Zoll*) ergibt 6 Druckzeilen pro Zoll Papierhöhe, während bei *8 lpi* 8 Druckzeilen auf ein Zoll (und damit natürlich auch mehr Zeilen auf eine Seite) passen.

Unter der Überschrift *Quality* können Sie (zumindest bei einigen Druckern) noch weiteren Einfluß auf die Druckqualität nehmen. Wählen Sie *Draft* (engl. für Entwurf) für schnelle Ausdrucke niedriger Qualität und *Letter* für (meist langsamere) Ausdrucke mit höherer Qualität. *Letter* steht dabei für *Near Letter Quality* (NLQ), wozu man im deutschen Sprachraum manchmal auch »Korrespondenzqualität« sagt.

Ob die Einstellungen unter den Punkten *Pitch*, *Quality* und *Spacing* etwas bewirken, hängt allerdings in starkem Maße von Ihrem Drucker ab. So haben manche Drucker nur eine Schriftqualität, andere nur eine oder zwei (nicht drei) Schriftweiten. Genauso bewirken manche Einstellungen bei einigen Druckern etwas beim Ausdruck von Texten, nicht aber beim Grafikdruck. Auch hier gilt – wie immer beim Anschluß zusätzlicher Geräte – Probieren geht über Studieren!

5.3.12 Einstellungen für Grafikdruck

Vom Druckeranpassungsfenster können Sie allerdings noch in ein weiteres Fenster von *Preferences* gelangen, das Sie vielleicht häufiger benötigen. Wenn Sie nämlich einen grafikfähigen Drucker besitzen, können Sie dort einstellen, wie die Darstellungen am Bildschirm auf das Papier umgesetzt werden. Wenn Sie nicht nur Texte, sondern auch einmal Bilder ausdrucken wollen, sollten Sie diese Einstellungen auf alle Fälle mindestens einmal vornehmen, oder noch besser, auch damit herumexperimentieren. Drücken Sie dazu im Druckeranpassungsfenster auf die Taste *Graphic Select* (engl. für *Grafikauswahl* oder *Grafikanpassung*).

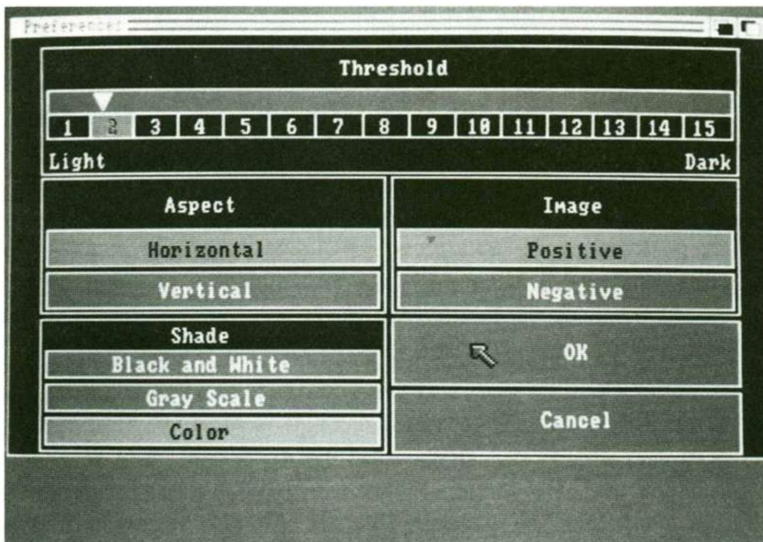


***Bild 5.12:** Einstellungen für den Grafikdruck*

In diesem Fenster können Sie insgesamt vier Einstellungen verändern: *Shade*, *Aspect*, *Image* und *Treshold*.

Bei *Shade* (engl. für *Schattierung* oder *Farbe*) haben Sie drei Möglichkeiten zur Wahl: *Black and White*, *Gray Scale* und *Color*. Die jeweils gültige davon wird (schwarz auf orange) hervorgehoben. *Black and White* (engl. für *schwarzweiß*) bedeutet, daß der Drucker beim Ausdruck jedes Bildschirm-Punktes die Entscheidung trifft, ob dieser schwarz oder weiß ausgedruckt wird. Welche Bildschirmfarben schwarz und welche weiß gedruckt werden, können Sie im Detail dann mit dem Schieberegler *Treshold* einstellen. Bei der Einstellung *Gray Scale* (engl. für *Graustufung*) versucht der Drucker, die verschiedenen Farben durch Grauschattierungen anzunähern. Hierbei werden aber mehrere Druckerpunkte pro Bildpunkt benötigt, weshalb bei einigen Druckern beziehungsweise einigen sehr detaillierten Bildern Schärfe verlorengeht und somit feine Details verschwimmen. Bei Bildern, bei denen es auf feine Details und Schärfe ankommt, sollten Sie also *Black and White* wählen und etwas mit der Justierung für *Treshold* experimentieren. Wenn Ihnen aber eine Annäherung der Bildschirmfarben wichtiger ist, müssen Sie *Gray Scale* wählen.

Die Einstellung *Color* (Farbe) ist nur dann sinnvoll, wenn Sie einen farbfähigen Drucker angeschlossen und in der Druckeranpassung korrekt ausgewählt haben. Die Farben auf dem Bildschirm erscheinen dann auch wirklich als verschiedene Farben auf dem Papier. Sie sollten jedoch selbst bei einem sehr guten und teuren Farbdrucker nicht erwarten, auch auf Papier ein ebenso brillantes Bild zu erhalten, wie es auf dem Bildschirm zu sehen ist. Die heutige Drucker-Technologie ist dazu leider in keiner Weise fähig. Zudem leuchtet der Bildschirm selbst, während die Farbe auf dem Papier nur Licht reflektiert, was immer einen anderen Effekt ergibt.

Aspect (Orientierung des Ausdrucks) bestimmt, ob ein Bild so ausgedruckt wird, wie es im Bildschirm zu sehen ist (*Horizontal*), ober ob es, um 90° gedreht, also quer, aus dem Drucker kommt (*Vertical*) – was bei großen Bildern, die breiter als hoch sind, durchaus sinnvoll sein kann.

Unter der Überschrift *Image* können Sie zwischen positivem (*Positive*) und negativem (*Negative*) Ausdruck wählen. Wird positiver Audruck gewählt, so werden helle Flächen bei einem Ausdruck in *Black and White* auch hell gedruckt (weiß oder hellgrau) und dunkle dunkel. Bei negativem Ausdruck werden schwarze und weiße (beziehungsweise hellgraue und dunkelgraue) Punkte ausgetauscht. Farbige Ausdrucke werden durch *Image* nicht beeinflußt.

Der Schieberegler *Treshold* (Schwarzweiß-Grenze) bestimmt beim Schwarzweiß-Druck, welche Farben weiß und welche schwarz ausgedruckt werden. Je näher der Knopf in diesem Schieberegler dem linken Anschlag kommt, desto weniger (nur sehr dunkle) Farben werden schwarz auf das Papier gedruckt, desto heller wird also das gesamte Bild. Je weiter der Regler nach rechts geschoben wird, desto hellere Farben werden noch schwarz ausgedruckt und um so weniger Farben werden weiß ausgegeben. Das Bild wird also insgesamt dunkler. Dies gilt zumindest, wenn die Einstellung für *Image* positiv ist. Ist sie negativ, bedeuten kleine Werte für *Treshold* ein dunkles Bild und große ein helles. Besitzen Sie einen grafikfähigen Schwarzweiß-Drucker, so müssen Sie wahrscheinlich eine Weile mit *Treshold* experimentieren, um

zufriedenstellende Ausdrucke von Bildern mit mehr als zwei Farben zu erreichen. Für unterschiedliche Bilder können sogar verschiedene *Treshold*-Werte nötig sein. Farbige Ausdrucke werden von *Treshold* nicht beeinflußt.

5.4 Für Verspielte: Das Werkzeug IconEd

Während *Preferences* ein sehr wichtiges Werkzeug ist, mit dem jeder Besitzer des Amiga 500 umgehen können sollte, ist das in diesem Abschnitt beschriebene Werkzeug mehr etwas für Verspielte: *IconEd*, ein Programm, mit dessen Hilfe Sie die Piktogramme (Icons), mit denen Disketten, Schubladen, Werkzeuge und Projekte auf der Diskette gezeigt werden, abändern (editieren) können.

Sie finden den Piktogramm-Editor in der Schublade *System* auf der Workbench-Diskette (bei einigen Versionen der Workbench befand er sich auch in der Schublade *Utilities*). Es ist ein relativ kleines Piktogramm mit dem Namen *IconEd*. Sie können dieses Werkzeug wie jedes andere auch mit einem Doppelklick auf das entsprechende Icon starten. Daraufhin erscheint zunächst ein Requester mit einigen wenigen (englischen) Informationen über das Programm. Klicken Sie auf den Knopf OK und dieser Requester verschwindet und gibt den Blick frei auf das Hauptfenster von *IconEd*.

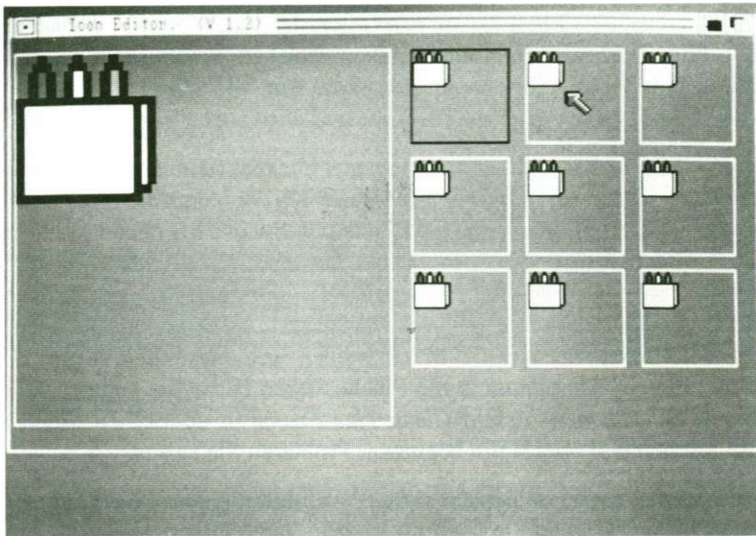


Bild 5.13: *Der Piktogramm-Editor*

Rechts sehen Sie neun kleine Quadrate, in denen sich das Piktogramm *IconEd* befindet. Sie werden später noch sehen, das Sie in jedes dieser Quadrate ein anderes Piktogramm laden können und dann schnell von der Bearbeitung eines Piktogramms zum nächsten springen können. Eines der neun kleinen Piktogramme ist etwas hervorgehoben, indem es einen anderen Rahmen besitzt (schwarz statt weiß, wenn Sie die Workbench-Farben nicht geändert haben). Dieses Piktogramm erscheint in vergrößerter Form links daneben (der Vergrößerungsfaktor beträgt etwa 4). In dieser vergrößerten Form können Sie das Piktogramm nun – ähnlich wie den Mauszeiger in *Preferences* – ändern. Der Mauszeiger wird dabei zu Ihrem Pinsel.

Zunächst aber müssen Sie eines der neun Piktogramme zur Bearbeitung auswählen. Dazu brauchen Sie nur in das entsprechende Quadrat zu klicken. Das ausgewählte Piktogramm wird dann schwarz umrahmt und erscheint in der Vergrößerung. Da nach dem Start alle neun Piktogramme gleich aussehen, bemerken Sie davon aber vielleicht nichts. Sie werden weiter unten noch erfahren, wie Sie andere Piktogramme von Diskette laden und ein beliebiges der neun Quadrate plazieren können.

5.4.1 Wie Sie im Piktogramm malen können

Sie können jedem Punkt des ausgewählten Piktogramms eine andere Farbe geben. Dazu brauchen Sie nur diesen Punkt anklicken. Das ist allerdings trotz der vergrößerten Darstellung nicht ganz leicht und verlangt einige Übung. Wenn Sie einen Punkt anklicken, erhält er die »aktuelle Farbe«. Klappen Sie einfach das Menü *Color* herunter, wenn Sie sehen wollen, was die aktuelle Farbe ist. Das Menü *Color* präsentiert die vier Workbench-Farben als schmale Streifen. Die aktuelle Farbe ist mit einem Haken versehen. Wenn Sie in einer anderen Farbe malen wollen, brauchen Sie diese andere Farbe nur wie einen Befehl aus dem Menü auszuwählen. Sie wird dann die aktuelle Farbe und wird auch abgehakt.

Sie müssen mit der Maus aber auch nicht jedesmal klicken, wenn Sie die Farbe eines Punktes ändern wollen. Wenn Sie die Selektionstaste innerhalb der Vergrößerung niederdrücken und festhalten, können Sie mit der Zeigerspitze wie mit einem Stift oder Pinsel malen. Gerade Linien, Kreise, Rechtecke und so weiter sind auf diese Weise aber nur sehr schlecht zu zeichnen. Entsprechende Hilfsmittel wie in einem richtigen Malprogramm fehlen *IconEd*. Sie können allerdings die Farbe einer größeren Fläche recht rasch ändern. Wählen Sie dazu die gewünschte neue Farbe aus dem *Color*-Menü, wählen Sie dann aus dem Menü *Misc* den Befehl *Flood Fill* und klicken Sie einmal in die Fläche, deren Farbe Sie ändern wollen. Genauso schnell können Sie mit dem Befehl *Clear This Frame* in demselben Menü das gesamte Piktogramm löschen und »bei Null« beginnen. Probieren Sie es aus:

Klicken Sie zunächst rechts im Fenster in ein Quadrat mit einem Piktogramm, das Sie noch nicht geändert haben. Wählen Sie dann die oberste Farbe (Blau) aus dem *Color*-Menü und rufen Sie den Befehl *Flood Fill* auf. Klicken Sie nun in der vergrößerten Darstellung in das Rechteck, das den größten Teil des Piktogramms von *IconEd* ausmacht, und mit einem Schlag ist es blau. Klicken Sie nun noch einmal in dieses Rechteck und halten Sie die Maustaste fest. Bewegen Sie die Maus langsam (!) nach rechts, bis eine durchgezogene blaue Linie den Rand des Piktogramms durchbricht. Füllen Sie nun das Piktogramm mit der Farbe Orange. Wählen

Sie dazu diese Farbe (die unterste) aus dem *Color*-Menü, wählen Sie *Flood Fill* und klicken Sie in die Fläche, die Sie eben blau gefärbt haben. Wie Sie sehen, läuft die Farbe bei *Flood Fill* sehr leicht aus!

5.4.2 Wie Sie Piktogramme verschieben, austauschen und mischen

Sie müssen für größere Änderungen aber nicht jeden Punkt des Piktogramms einzeln bearbeiten. *IconEd* bietet auch einige Befehle für größere Manipulationen an Piktogrammen. Mit dem Befehl *In Frame* im Menü *Move* können Sie das Piktogramm zum Beispiel als Ganzes bewegen. Wenn Sie diesen Befehl aus dem Menü auswählen, erscheint rechts statt der neun kleinen Piktogramme eine Art Requester. In dessen oberer Hälfte befinden sich vier Pfeile. Sie können nun das Piktogramm in der Vergrößerung in eine beliebige Richtung schieben, indem Sie auf den Pfeil klicken, der in diese Richtung zeigt. Wenn Sie auf das Quadrat in der Mitte der vier Pfeile klicken, springt das Piktogramm wieder dahin zurück, wo es am Anfang lag.

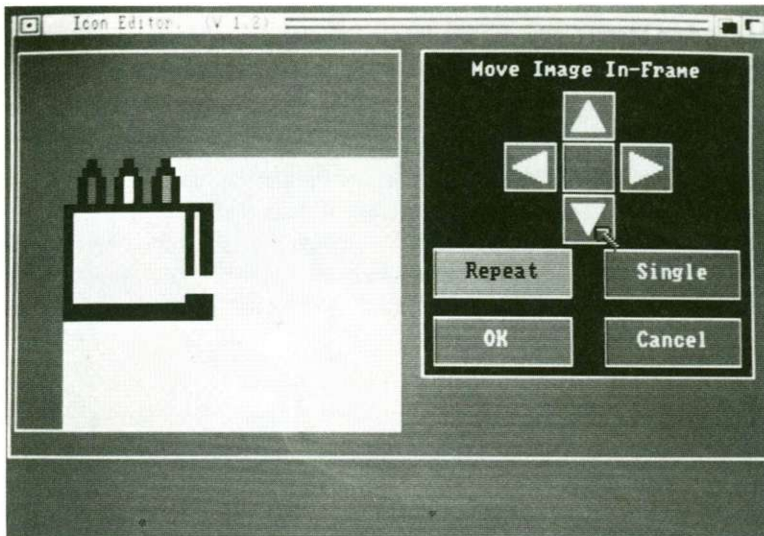


***Bild 5.14:** Verschieben eines Icons*

Normalerweise bewegt sich das Piktogramm bei jedem Mausklick auf einen der vier Pfeile nur einen Punkt in die angegebene Richtung. Sie können diesen Vorgang beschleunigen, indem Sie auf den Knopf *Repeat* (engl. für *wiederholen*) darunter klicken. Die Bewegung des Piktogramms hält dann, nachdem Sie auf einen der Pfeile geklickt haben, so lange an, wie Sie die Selektionstaste der Maus festhalten. Wenn Sie den alten Zustand (der schrittweisen Bewegung) wiederherstellen wollen, brauchen Sie nur einmal auf den Knopf *Single* zu klicken.

Bei jedem Mausklick bewegt sich das Piktogramm dann wieder nur einen Punkt in die Richtung, in die der angeklickte Pfeil weist.

Sie müssen den Verschiebevorgang schließlich mit einem Klick auf *OK* oder *Cancel* abbrechen. *OK* bestätigt die Verschiebung, während ein Klick auf *Cancel* den alten Zustand des Piktogramms vor dem Aufruf von *Move In-Frame* unverändert beläßt.

Neben dieser Verschiebung bietet das *Move*-Menü noch einen zweiten Befehl. *Exchange With Frame* bringt ein Submenü zum Vorschein. Wenn Sie aus diesem Menü eine Zahl wählen, wird das gerade bearbeitete Piktogramm mit einem anderen der neun Piktogramme ausgetauscht. Die Anordnung der neun Zahlen im Submenü *Exchange With Frame* entspricht dabei exakt den Positionen der Piktogramme in den neun Quadraten in der rechten Fensterhälfte.

Genauso einfach können Sie auch ein Piktogramm durch ein anderes ersetzen (hierbei geht Ihnen im Gegensatz zum gerade beschriebenen *Exchange With Frame* eines des neun Piktogramme verloren). Wählen Sie dazu zuerst dasjenige der neun kleinen Piktogramme durch Anklicken aus, das Sie ersetzen wollen. Gehen Sie dann mit der Maus auf den Befehl *From Frame* aus dem Menü *Copy*, der wieder ein Submenü zum Vorschein bringt. Wählen Sie aus diesem Submenü dann die Nummer des Piktogramms, das das eben ausgewählte ersetzen soll.

Eine interessante Alternative zum Kopieren eines ganzen Piktogramms erlaubt es Ihnen, Piktogramme zu »mischen«, das heißt, die eingefärbten Punkte von zwei Piktogrammen zusammenzulegen. Klicken Sie dazu zuerst auf dasjenige der neun kleinen Piktogramme, das eines der beiden zu mischenden Piktogramme enthält. Wählen Sie dann aus dem *Copy*-Menü *Merge with Frame* und aus dem daraufhin auftauchenden Submenü die Nummer des anderen Piktogramms. Diese beiden Piktogramme »zusammen« ersetzen dann das aktuelle Piktogramm (mit dem schwarzen Rahmen).

Die restlichen beiden Befehle im *Copy*-Menü ermöglichen es, Ihre eigenen Fehler wieder rückgängig zu machen; oder besser: es sich nach einer größeren Änderung noch einmal anders zu überlegen. Wählen Sie dazu vor jeder größeren Änderung an einem komplizierten Piktogramm den Befehl *Snapshot Frame* aus dem Menü *Copy*. *IconEd* legt dann eine exakte Kopie des aktuellen Piktogramms (mit dem schwarzen Rahmen) in einen Pufferspeicher. Führen Sie nun Ihre Änderungen aus. Gefallen Ihnen die Ergebnisse Ihrer Bemühungen nicht, können Sie die letzte Version des Piktogramms nun mit *Undo Frame* wieder »zurückholen«. Jedesmal, wenn Sie *Undo Frame* aus dem Menü *Copy* wählen, ersetzt das mit *Snapshot Frame* abgespeicherte Piktogramm das gerade bearbeitete – auch wenn sich dieses an einer anderen Position befindet als das mit *Snapshot Frame* abgespeicherte!

5.4.3 Wie Sie kurze Texte in ein Piktogramm setzen

IconEd besitzt auch eine Möglichkeit, kurze Texte in ein Piktogramm zu plazieren. Das kann zum Beispiel ganz praktisch sein, wenn Sie ein Piktogramm erstellen wollen, das ein oder zwei verfremdete Buchstaben enthält. Ansonsten sollten Sie mit Texten in Piktogrammen sparsam umgehen. Zum Beispiel sollte der Name eines Piktogramms nie im Piktogramm selbst stehen.

Er wird von der Workbench ja daruntergesetzt und kann mit dem *Rename*-Befehl geändert werden.

Wenn Sie aber trotzdem einen Text in ein Piktogramm setzen wollen, brauchen Sie nur den Befehl *Write Into Frame* aus dem Menü *Text* aufzurufen. Daraufhin wird das rechte Drittel des Bildschirms von einer Art Requester eingenommen. Sie müssen hier nun zunächst im Text-Gadget am oberen Rand (direkt unter *Icon Text*) den gewünschten Text eintragen. In den beiden Feldern darunter können Sie die Vorder- und Hintergrundfarbe (*Foreground* und *Background*) wählen, mit der Text im Piktogramm erscheinen wird. Sie können diese beiden Farben einfach ändern, indem Sie in das entsprechende farbige Rechteck neben *Foreground* oder *Background* klicken. Im Feld *Font* (Schriftart) können Sie nur die Schriftgröße wählen, obwohl der Name *Font* auch eine Wahlmöglichkeit der Schriftart suggeriert. Ihnen steht nur die Schriftart *Topaz* (Sie kennen sie aus dem *Notepad*) in den Größen 8 und 9 zur Verfügung. Die Größe 8 heißt hier *Topaz_Eighty* und die Größe 9 *Topaz_Sixty*. Sie können auch wieder von einer zur anderen Größe wechseln, indem Sie auf den Schriftnamen klicken (*Topaz_Eighty* oder *Topaz_Sixty*).

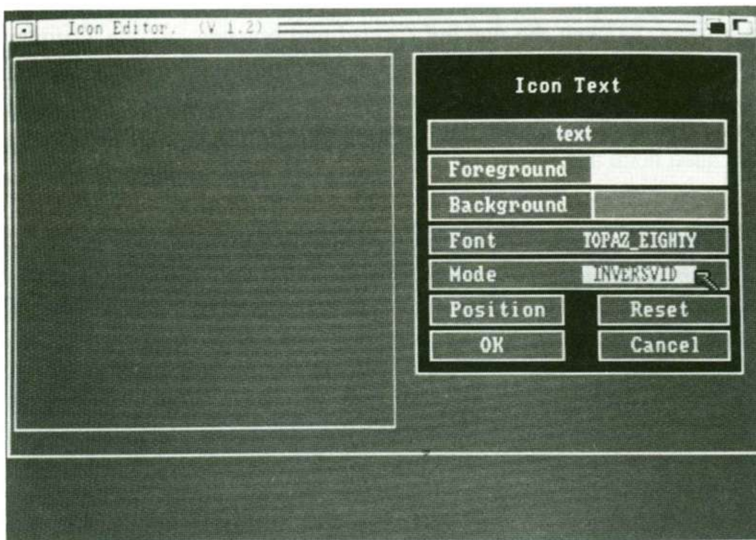


Bild 5.15: Einsetzen eines Textes in einem Piktogramm

Nun kommt noch die Wahl des *Modus* (engl. *mode*), der bestimmt, wie die neuplazierte Schrift auf die Punkte im Piktogramm wirkt, die dort schon stehen, wo die Schrift erscheinen soll. Ihnen stehen vier verschiedene Einstellungen für Mode zur Verfügung: *JAM1*, *JAM2*, *COMPLEMENT* und *INVERSVID*. Im Modus *JAM1* erscheinen nur die Punkte der Buchstaben selbst (in der Vordergrundfarbe) im Piktogramm und ersetzen die Punkte, die vorher dort

standen. Im Modus *JAM2* erscheinen sowohl die Punkte der Buchstaben (der »Vordergrund«) im Piktogramm als auch das Buchstabenumfeld (der »Hintergrund«) in der gewählten Hintergrundfarbe. Beide Gruppen von Punkten ersetzen (überdecken) die Punkte, die vorher dort standen. Im Modus *COMPLEMENT* werden die Punkte der Buchstaben (ohne Hintergrund) mit den schon im Piktogramm stehenden Punkten kombiniert. Diesen Modus verstehen Sie am einfachsten durch Ausprobieren (siehe unten)! Und im Modus INVERSVID erscheint nur das Umfeld der Buchstaben in der eingestellten Hintergrundfarbe im Piktogramm. Die Buchstaben selbst sind »durchscheinend« und lassen die alten Punkte des Piktogramms stehen.

Am besten probieren Sie die verschiedenen Modi nun, damit Sie ein Gefühl dafür bekommen. Rufen Sie dazu *Write Into Frame* aus dem Menü *Text* auf und geben Sie einen kurzen Text ein. Ändern Sie dann gegebenenfalls die beiden Farben (keine der Farben sollte gleich der Farbe des Bildschirmhintergrunds sein) und klicken Sie auf den Knopf *Position* (engl. *to position = positionieren*). Es erscheint nun ein Requester, der dem ähnlich sieht, mit dem Sie auch das Piktogramm im Rahmen verschieben können (*Move In-Frame*). Gleichzeitig erscheint der eingegebene Text im Piktogramm. Sie können ihn nun mit den vier Pfeilen bewegen. Dabei können Sie verfolgen, wie die Punkte der Buchstaben beziehungsweise ihr Umfeld auf das alte Piktogramm wirken. Klicken Sie nun auf *Cancel* und brechen Sie den Vorgang damit ab. Ändern Sie nun den Modus (*Mode*) und klicken Sie erneut auf Position. Bewegen Sie nun wieder den Text über das Icon und beachten Sie die Wirkung von Buchstaben und Buchstabenhintergrund. Klicken Sie dann wieder auf *Cancel* und probieren Sie die beiden anderen Modi auch noch aus.

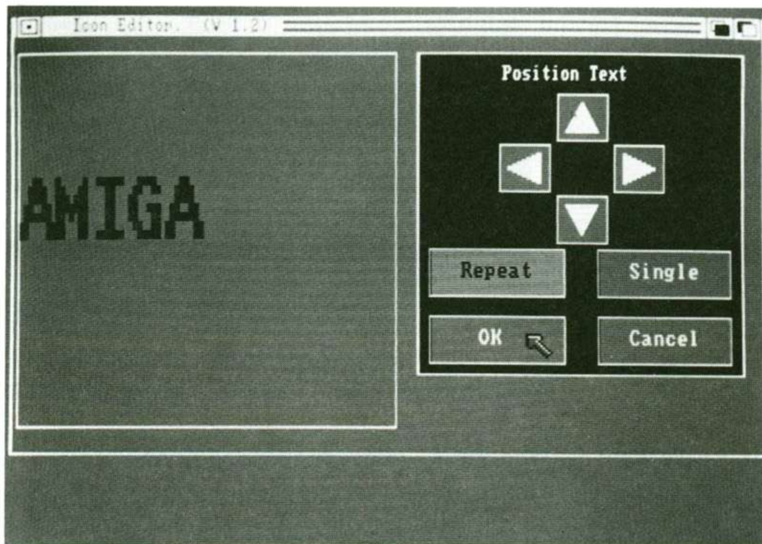


***Bild 5.16:** Positionieren eines Textes in einem Piktogramm*

Wenn Sie den Requester *Position Text*, der nach dem Klick auf *Position* erscheint, mit einem Klick auf OK beenden, bleibt der Text an genau der Stelle im Piktogramm stehen, wo er zuletzt zu sehen war. Auf diese Weise plazieren Sie also den vorher eingegebenen Text im Piktogramm und kehren zurück zum Requester mit dem Titel *Icon Text*. Falls Ihnen der Text dann nicht oder an dieser Stelle nicht gefallen sollte, können Sie diese Änderung nun immer noch mit einem Klick auf *Cancel* im Requester *Position Text* rückgängig machen und unverrichteter Dinge zum Hauptfenster zurückkehren. Nur wenn Sie im Requester *Position Text* auf OK klicken, werden die Eingaben und Änderungen, die Sie nach dem Aufruf des Befehls *Write Into Frame* gemacht haben, wirksam.

5.4.4 Wie Sie ein Piktogramm zum Ändern einlesen

So, nun wissen Sie schon, wie Sie alle möglichen Dinge mit einem Piktogramm anstellen können, aber noch nicht, wie Sie beliebige Piktogramme von der Diskette einlesen, damit Sie überhaupt etwas zum Ändern haben. Das werde ich deshalb nun schleunigst nachtragen.

Bevor Sie ein Piktogramm einlesen, klicken Sie zuerst auf eines der neun kleinen Quadrate in der rechten Bildschirmhälfte. In diesem Quadrat wird das neu eingelesene Piktogramm erscheinen. (Sie können auf diese Weise mit bis zu neun verschiedenen Piktogrammen gleichzeitig arbeiten.) Wählen Sie dann den Befehl *Load Data* aus dem Menü *Disk*. Daraufhin erscheint ein Requester in der rechten Bildschirmhälfte, der die Überschrift *Load Icon Image Data* trägt. Er enthält am oberen Rand ein Text-Gadget, in das Sie den Namen des Piktogramms eintragen müssen, das Sie einlesen wollen. (Sie können den alten Namen, der schon dort steht, mit der Tastenkombination [Ctrl]+[X] löschen.)

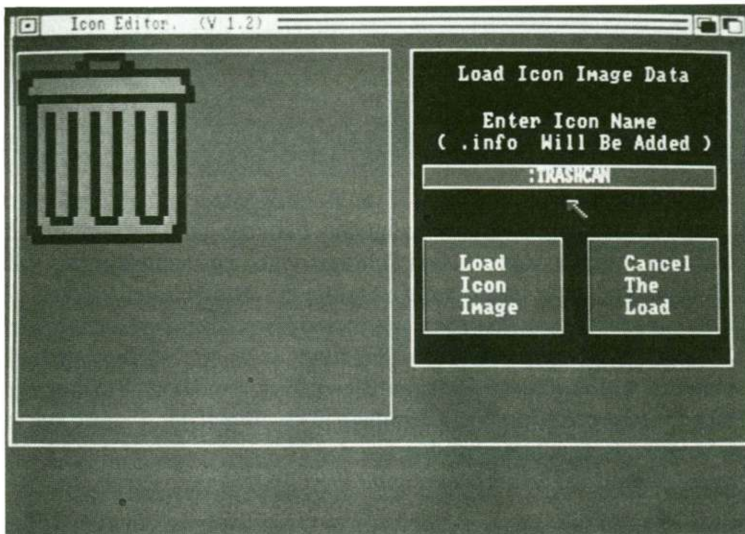


***Bild 5.17:** Einlesen eines Icons*

In den meisten Fällen genügt es jedoch nicht, hier nur den Namen einzutragen, der unter dem Piktogramm steht. Sie müssen *IconEd* nämlich auch mitteilen, in welcher Schublade sich das Piktogramm befindet. Nur wenn sich das Piktogramm in derselben Schublade befindet wie auch *IconEd* selbst, reicht der Name des Piktogramms.

Für Piktogramme, die in anderen Schubladen liegen, müssen Sie zunächst einen Doppelpunkt (:) eingeben und dann hintereinander die Namen der Schubladen angeben, die geöffnet werden müssen, um das gewünschte Piktogramm zu finden. Diese Namen müssen durch Schrägstriche getrennt werden. Auch hinter dem Namen der letzten Schublade muß noch ein Schrägstrich gesetzt werden und dahinter schließlich der Name des gewünschten Piktogramms. Um das Piktogramm *Müller* in der Schublade *Briefe*, die selbst wieder in der Schublade *Texte* liegt, einzulesen, müßten Sie in diesem Textrequester also den Namen *:Texte/Briefe/Müller* eintragen. (Das ist ein sogenannter »Pfadname«, über den Sie im übernächsten Kapitel *Einführung in das CLI* noch mehr erfahren werden.)

Auf diese Art können Sie das Piktogramm jeder Schublade, jedes Werkzeugs, Projektes und des Mülleimers einlesen. Die Schublade Briefe aus dem obigen Beispiel hätte zum Beispiel den Namen *:Texte/Briefe* und der Mülleimer auf derselben Diskette den Namen *:Trashcan*. Wenn Sie jedoch das Piktogramm einer Diskette modifizieren wollen, müssen Sie etwas anders vorgehen. Das Piktogramm der Diskette, auf der sich der Piktogrammeditor selbst befindet, trägt den Namen *:Disk* (achten Sie auf den Doppelpunkt am Anfang). Das Piktogramm einer anderen Diskette hat den Namen *<Diskettenname>:Disk*, wobei Sie <Diskettenname> durch den Namen ersetzen müssen, der unter dem Piktogramm dieser Diskette auf der Workbench erscheint.

Wenn Sie den Namen des Piktogramms eingetragen haben, können Sie auf den Knopf *Load Icon Image* klicken (wenn Sie es sich anders überlegt haben, auf den Knopf *Cancel The Load*). Das neue Piktogramm sollte dann im zuvor angeklickten Quadrat und in der Vergrößerung in der linken Bildschirmhälfte erscheinen. Sie können es nun nach Herzenslust ändern.

Wenn Sie bei der Eingabe des Namens einen Fehler gemacht haben, erscheint allerdings kein neues Piktogramm. Statt dessen steht in der Titelzeile *Icon Editor - Loading error (Press mouse button)*. Sie sollten dann noch einmal *Load Data* im *Disk*-Menü aufrufen und versuchen, den richtigen Namen, wie oben beschrieben, einzugeben. Falls Sie nicht verstanden haben, wie der lange Name mit dem Doppelpunkt und den Schrägstrichen zustande kommt, sollten Sie Ihre *IconEd*-Experimente vielleicht erst einmal abschließen und Kapitel 7 lesen. Eine andere Alternative ist es, zunächst einmal *IconEd* zu verlassen (indem Sie das Hauptfenster schließen) und dann das Piktogramm *IconEd* in dieselbe Schublade zu legen, in dem auch das Piktogramm liegt, das Sie einlesen wollen. Oder Sie können dieses Piktogramm auch in dieselbe Schublade legen, in der sich *IconEd* befindet. Wenn Sie dann noch einmal den *IconEd* starten und *Load Data* im *Disk*-Menü aufrufen, brauchen Sie nur den Namen im Requester *Load Icon Image Data* einzutragen, der auf der Workbench unter dem Piktogramm erscheint, das Sie ändern wollen.

5.4.5 Wie Sie ein Piktogramm abspeichern

Wenn Sie alle Änderungen an einem Piktogramm abgeschlossen haben, wollen Sie es natürlich auch wieder abspeichern, beziehungsweise gegen das alte Piktogramm ersetzen. Klicken Sie in diesem Fall auf dasjenige der neun kleinen Piktogramme in der rechten Fensterhälfte, das Sie abspeichern wollen, und rufen Sie dann den Befehl *Save Data* im Menü *Disk* auf. Es erscheint ein Requester, der dem sehr ähnlich sieht, der nach *Load Data* auftaucht. Auch hier müssen Sie wieder in dem Text-Gadget am oberen Rand den Namen eintragen, unter dem das Piktogramm gesichert werden soll. Wie Sie solche Namen für Piktogramme bilden, die nicht in derselben Schublade liegen wie der *IconEd*, können Sie im letzten Abschnitt nachlesen. Achten Sie sehr sorgfältig auf Tippfehler, sonst kann es Ihnen passieren, daß Sie ein neues Piktogramm abspeichern, obwohl Sie eigentlich ein altes überschreiben wollten.

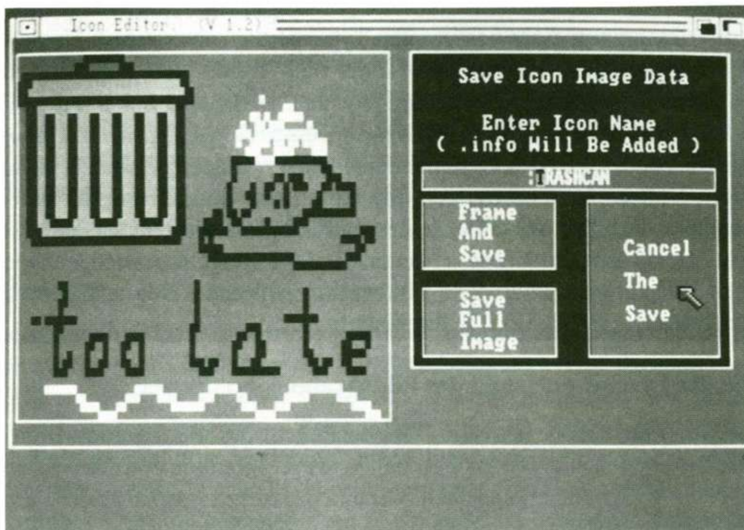


Bild 5.18: *Abspeichern eines Icons*

Wenn Sie den Namen eingetragen haben, können Sie das Piktogramm mit einem Klick auf *Save Full Image* oder *Frame And Save* auf die Diskette schreiben. Wenn Sie auf *Save Full Image* klicken, wird das Piktogramm in seiner vollen Größe, wie es im Rahmen rechts daneben gezeigt wird, abgespeichert. Dies ist meist nicht das, was Sie wollen. Das »eigentliche Piktogramm« macht ja meist nur einen kleinen Teil des gesamten Quadrats aus, das bearbeitet werden kann. Klicken Sie in solchen Fällen auf den Knopf *Frame And Save* (zu deutsch etwa *umrahmen und dann sichern*). Sie können nun mit der Maus den Bereich markieren, der wirklich als Piktogramm abgespeichert werden soll. Klicken Sie dazu in der Vergrößerung einmal in die linke obere Ecke des Rechtecks, das das »eigentliche Piktogramm« vollständig

umschließt. Gehen Sie dann mit der Mausspitze in die rechte untere Ecke dieses Rechtecks. Ein dünner Rahmen markiert daraufhin die gewünschte Größe. Wenn er genau den Bereich umschließt, den Sie abspeichern wollen, müssen Sie ein zweites Mal auf die Selektionstaste drücken. Das Piktogramm wird dann abgespeichert. Wenn Sie *IconEd* nun verlassen wollen, brauchen Sie nur das Hauptfenster zu schließen (mit dem Schließ-Gadget).

Falls es aus irgendeinem Grund nicht möglich ist, das Piktogramm unter dem von Ihnen angegebenen Namen abzuspeichern, erscheint wieder eine Fehlermeldung in der Titelleiste des Fensters. Die häufigsten Gründe dafür sind der aktivierte Schreibschutz einer Diskette und ein Tippfehler in dem langen Namen eines Piktogramms aus einer anderen Schublade.

Sie können beim Ändern von Piktogrammen aber noch ein paar andere Fehler machen, die nicht so leicht zu entdecken sind. Sie können diese Fehler leicht vermeiden, wenn Sie die folgenden Punkte beachten:

Sichern Sie ein Piktogramm eines bestimmten Typs niemals unter einem Namen ab, der zu einem anderen Typ gehört! Es gibt insgesamt fünf Typen von Objekten, denen fünf verschiedene Arten von Piktogrammen entsprechen: Disketten, Schubladen, Mülleimer, Projekte und Werkzeuge. Wenn Sie ein Piktogramm eines dieser Typen von Objekten ändern wollen, sollten Sie zunächst ein Piktogramm desselben Typs laden (vielleicht sogar eines, das Sie auch ändern wollen), es modifizieren und dann unter demselben oder einem anderen Namen abspeichern. Der Typ des Objekts, dessen Piktogramm Sie laden, sollte immer dem Typ des Objekts gleich sein, unter dessen Namen Sie das Piktogramm wieder abspeichern. Sie dürfen also zum Beispiel kein Schubladen-Piktogramm einlesen und es dann unter dem Namen eines *Notepad*-Projektes (Notizblocks) abspeichern.

5.4.6 Wie Sie das Erscheinungsbild des Piktogramms bestimmen

Damit haben Sie nun schon fast alle Möglichkeiten des Piktogramm-Editors *IconEd* kennengelernt. Sie können zum Beispiel vorhandene Piktogramme von der Diskette einlesen, abändern und dann wieder unter demselben oder einem anderen Namen abspeichern. *IconEdit* bietet zusätzlich noch zwei Befehle, mit denen man das Erscheinungsbild selbsterstellter oder modifizierter Piktogramme auf der Workbench abändern kann.

Die erste dieser Möglichkeiten verbirgt sich hinter dem Befehl *Set Bottom Border* im Menü *Misc*. Dies ist ein Submenü, das Ihnen die beiden Möglichkeiten *0* und *1* zur Wahl bietet. Die gerade gültige Einstellung ist wie üblich mit einem Haken versehen. Wenn Sie *1* wählen und dann das gerade bearbeitete Piktogramm abspeichern, läßt die Workbench zwischen dem unteren Rand des Piktogramms und dem darunterstehenden Namen eine dünne Linie frei. Das sieht besonders bei Piktogrammen mit einer geraden unteren Kante besser aus. Wählen Sie hingegen im Submenü *Set Bottom Border* Möglichkeit *0*, so geht der Text des Namens bis an den unteren Rand des Piktogramms.

Das Menü *HiLite* (engl. *to hilite = aufhellen/hervorheben*) bietet schließlich noch die Wahl zwischen *Inverse* und *Backfill*. Ihre Wahl in diesem Menü bestimmt, wie ein Piktogramm, das Sie abgespeichert haben, auf der Workbench hervorgehoben wird, wenn Sie es anklicken. Wählen Sie *Inverse*, wird es invertiert, das heißt, die vier Farben, aus denen es besteht, werden

untereinander ausgetauscht. Genau dasselbe passiert auch bei *BackFill*. Im Unterschied zu *Inverse* ändern bei *Backfill* jedoch nicht die Gebiete eines Piktogramms ihre Farbe, die erstens die Hintergrundfarbe (die vierte im *Color*-Menü) tragen und zweitens an den Rand des Rechtecks grenzen, das das Piktogramm umschließt. Das sieht aus, als würde dieser Rand in der Hintergrundfarbe gar nicht zum Piktogramm gehören (soll er meist auch nicht) und ist wohl die am häufigsten verwendete Art, ein Piktogramm hervorzuheben.

Wenn Sie diese beiden Arten der Hervorhebung einmal direkt vergleichen wollen, brauchen Sie nur die Systemschublade zu öffnen. Das Piktogramm *SetMap* darin verwendet zur Hervorhebung die *Inverse*-Methode, während *SetMap* selbst und auch *NoFastMem* die *Backfill*-Methode verwenden.

5.5 »Kleine« Werkzeuge in der Systemschublade

Neben *IconEd* enthält die Systemschublade auf der Workbench noch eine Unzahl »kleiner« Werkzeuge, die Sie zum Teil vielleicht einmal gebrauchen werden. Viele davon werden Sie aber vielleicht nur ein- oder zweimal starten und dann wieder vergessen. Das heißt aber nicht, daß diese Werkzeuge unwichtig wären. Manche davon werden in gewissen Situationen automatisch gestartet, ohne daß Sie etwas davon merken. Löschen Sie sie also bitte nicht, ohne zuvor ihre Bedeutung verstanden zu haben.

5.5.1 SetMap

Das Werkzeug *SetMap*, das sich in der Systemschublade befindet, werden Sie wahrscheinlich relativ selten benötigen. Es dient dazu, die »Tastaturbelegung« (engl. *keymap*) des Amiga 500 zu ändern. Diese Tastaturbelegung bestimmt, welcher Buchstabe auf dem Bildschirm erscheint, wenn Sie eine bestimmte Taste drücken. Es ist nämlich keineswegs selbstverständlich, daß ein »a« auf dem Notizblock erscheint, wenn Sie *Notepad* starten und auf die Taste [A] drücken. Wenn Sie eine Taste betätigen, geht eine Folge elektrischer Impulse von einem bestimmten Chip im Amiga 500 an einen anderen. Wie diese Impulse interpretiert werden, also die Zuordnung der Impulsfolge zu einem bestimmten Buchstaben, bestimmt die Systemsoftware des Amiga 500. Sie schaut dazu in einer Tabelle nach, der *keymap*. Diese *keymap* können Sie mit *SetMap* (engl. *map = Karte*) »setzen«, also verändern.

Sie werden sich nun vielleicht fragen, wozu denn das gut sein soll. Schließlich wollen Sie ja, daß ein »a« erscheint, wenn Sie die Taste [A] drücken. Die Antwort darauf ist relativ einfach: Der Amiga 500 wird nicht nur nach Deutschland verkauft und in anderen Ländern ergeben dieselben Tasten andere Buchstaben. Die beiden Tasten, die bei uns das »Ö« und das »Ä« ergeben, dienen auf einem englischen Amiga zur Eingabe der eckigen Klammern (die englische Sprache kennt ja keine Umlaute).

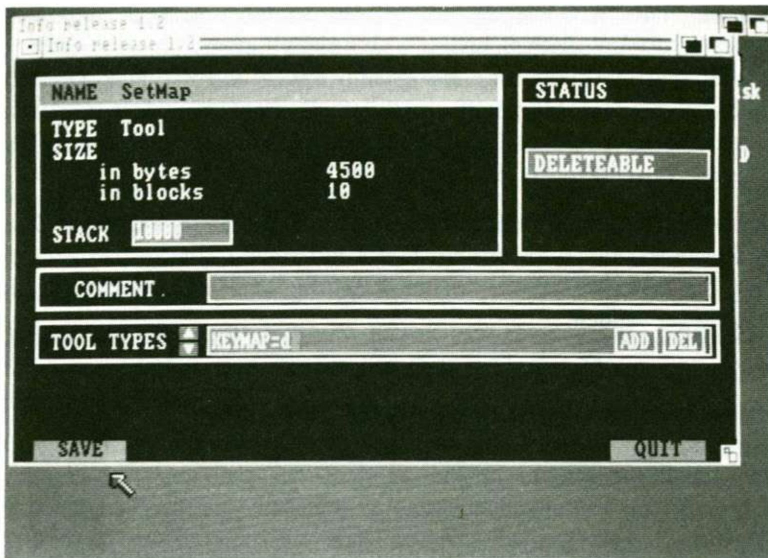


Bild 5.19: Das Info-Fenster von SetMap

Mit *SetMap* können Sie auf die Tastaturbelegung eines anderen Landes umschalten. Selektieren Sie hierzu das Piktogramm *SetMap* und wählen Sie *Info* aus dem Menü *Workbench*. Daraufhin erscheint das Info-Fenster von *SetMap*. Klicken Sie nun einmal auf den Knopf *Add* am rechten Fensterrand (auf gleicher Höhe mit den Worten TOOL TYPES). Klicken Sie dann in das Text-Gadget TOOL TYPES und geben Sie den folgenden Text ein »KEYMAP=<xx>« (natürlich ohne die Anführungsstriche), wobei Sie <xx> durch das Kürzel eines Landes ersetzen müssen, dessen Tastaturbelegung Sie wünschen:

Kürzel	**Land**
d	Deutschland/Österreich/Schweiz
e	Spanien
f	Frankreich
gb	Großbritannien
i	Italien
is	Island
s	Schweden
usa	USA
usa0	USA (alt; wie Workbench Version 1.1)
usa2	USA Dvorak-Belegung

Tabelle 5.1: Länderkürzel für SetMap

Probieren Sie zum Beispiel einmal »KEYMAP=gb« aus. Schließen Sie die Eingabe im Feld TOOL TYPES durch Betätigung von [Return] ab und klicken Sie dann auf den Knopf SAVE unten links im Info-Fenster. Wenn Sie nun *SetMap* (zum Beispiel mit einem Doppelklick auf das Piktogramm) starten, geschieht scheinbar nichts. Wenn Sie dann aber *Notepad* starten und ein paar Sätze eingeben, werden Sie feststellen, daß die Beschriftung der Tasten nicht mehr mit den Buchstaben übereinstimmt, die sie ergeben. Sie arbeiten nun mit einer englischen Tastatur!

Sie können das aber leicht wieder rückgängig machen. Die brutalste Methode ist zweifellos ein Neustart. Eleganter ist es natürlich, im Feld TOOL TYPES von *SetMap* »KEYMAP=gb« durch »KEYMAP=d« zu ersetzen und *SetMap* noch einmal zu starten. Sie brauchen dazu nicht einmal *Notepad* zu verlassen.

Löschen Sie *SetMap* auch dann nicht, wenn Sie meinen, es nicht zu brauchen. Das Programm wird beim Neustart automatisch aufgerufen, um die deutsche Tastenbelegung zu aktivieren. Wenn *SetMap* dann nicht vorhanden ist, behält der Amiga 500 die voreingestellte (amerikanische) Tastenbelegung.

5.5.2 Graphic Dump

Das Werkzeug *Graphic Dump* dient dazu, den Bildschirminhalt, wie Sie ihn in einem bestimmten Moment sehen, auf einem (grafikfähigen!) Drucker auszugeben. Dieser Drucker muß dazu angeschlossen, eingeschaltet und mit *Preferences* zuvor »angemeldet« worden sein. Computerfachleute sagen zu einem solchen Bildschirmausdruck »Dump« oder »Screendump«. Da es sich beim Amiga immer um eine Bildschirmgrafik handelt, spricht man von einem »Graphic Dump«, also grafischen Screendump.

Wenn Sie einen solchen Screendump ausdrucken wollen, brauchen Sie *Graphic Dump* nur zu starten (mit einem Doppelklick auf das Piktogramm). Nach dem Start bleiben Ihnen noch ungefähr 10 Sekunden, den Bildschirm so zu gestalten, daß Sie das sehen, was ausgedruckt werden soll, dann beginnt *Graphic Dump* zu arbeiten (drucken). Sie können in dieser Zeit zum Beispiel mit [*C*=]+[M] schnell den Workbench-Screen nach hinten legen. Ausgedruckt wird dann der in diesem Moment vorne liegende Screen, ohne einen eventuell darüber liegenden Mauszeiger. (Es gibt keinen Weg, den Mauszeiger auszudrucken.) Wenn in diesem Moment jedoch ein Menü sichtbar ist, wird es mit eventuell hervorgehobenen Menüpunkten (aber ohne Mauszeiger) ausgedruckt.

Ist der Drucker nicht oder nicht korrekt angeschlossen und mit *Preferences* angemeldet worden, erscheint nach Ablauf von etwa 20 Sekunden eine Fehlermeldung. Klicken Sie in dem dabei auftauchenden Requester dann einfach auf CANCEL, schalten Sie den Drucker ein (oder beheben den Fehler auf andere Weise) und versuchen Sie es noch einmal.

Bild 5.20: Fehlermeldung bei nicht eingeschaltetem Drucker

5.5.3 Say

Das Werkzeug *Say* (engl. to say = *sagen/sprechen*) dient zur Demonstration der Sprachsynthese, zu der der Amiga 500 fähig ist. Sie können Texte eingeben und dann zuhören, wie Sie der Amiga 500 »ausspricht«. Starten Sie dazu *Say* (mit einem Doppelklick auf das Piktogramm) und warten Sie ab, bis zwei neue Fenster erschienen sind. Das eine heißt *Input Window* (Eingabe-Fenster), das andere *Phoneme Window* (Phonem-Fenster).

Im Eingabe-Fenster tippen Sie die Sätze ein, die der Amiga aussprechen soll. (Klicken Sie vorher einmal mit der Maus in dieses Fenster.) Jedesmal, wenn Sie einen Satz mit [Return] abschließen, wird er ausgesprochen und es erscheinen parallel dazu die »Phoneme«, aus denen er besteht, im Phonem-Fenster. Phoneme sind die Bausteine, aus denen gesprochene Wörter bestehen. Ein Phonem entspricht einer Silbe. Die »Phonemschrift« ist im Gegensatz zu allen anderen Schriften nicht sprachabhängig. Das heißt, die gleiche Folge von Phonemen wird – abgesehen von einem leichten Akzent – in allen Sprachen gleich ausgesprochen. Der Amiga 500 hat übrigens – entsprechend seiner Herkunft – einen amerikanischen Akzent.

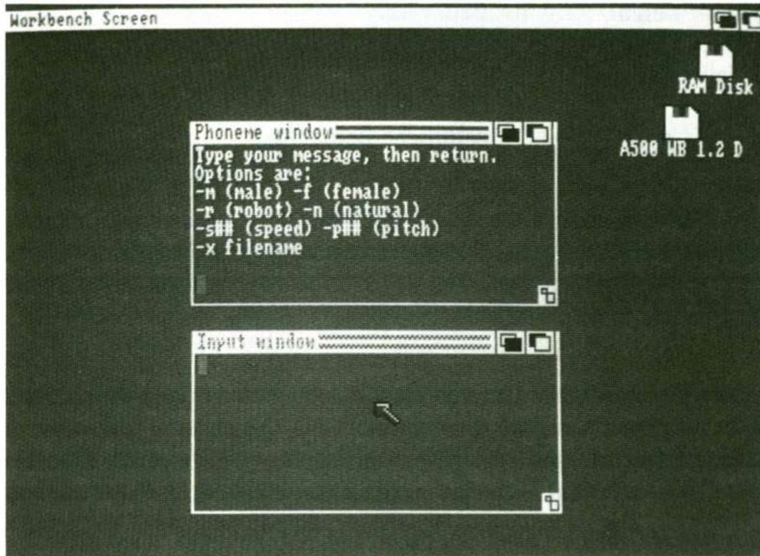


Bild 5.21: Die beiden Fenster des Sprachausgabeprogramms Say

Welches Phonem wie ausgesprochen wird, erfahren Sie im AmigaBASIC-Handbuch. Sie müssen das aber nicht unbedingt wissen, um *Say* sprechen zu lassen. Englische Sätze bewältigt *Say* sehr gut. Und wenn Sie Ihren Amiga deutsch sprechen lassen wollen, so überlegen Sie sich einfach, welches englische Wort so ausgesprochen würde wie das gewünschte deutsche. Den Satz »der mowrer hoult den hummer« spricht *Say* zum Beispiel als »Der Maurer holt den Hammer« aus.

Neben Sätzen, die der Amiga aussprechen soll, können Sie aber auch noch ein paar Befehle eingeben, die die Aussprache des Amiga 500 beeinflussen. Hierzu müssen Sie einen neuen Satz einfach mit einem Bindestrich (Minus) beginnen. »-f« (für *female = weiblich*) schaltet auf eine weibliche Stimme um, »-m« (für *male = männlich*) auf eine männliche. Mit »-n« (für natürlich) erreicht man eine einigermaßen normale Aussprache, wobei der Amiga 500 seine Stimme hebt und senkt und verschiedene Silben verschieden stark betont. Dagegen schaltet »-r« auf eine roboterhafte Stimme ohne jede Betonung um. Mit den beiden Befehlen »–p« (für *pitch = Tonhöhe*) und »–s« (für *speed = Geschwindigkeit*) kann man Tonhöhe und Geschwindigkeit der Aussprache ändern. Hierzu muß hinter »-p« eine Zahl zwischen 65 (sehr tief/dunkel) und 320 (sehr hoch/hell) und hinter »-s« eine Zahl zwischen 40 (sehr langsam) und 400 (sehr schnell) folgen. Die Zahlen müssen direkt, ohne eine Leerstelle dazwischen, hinter »p« oder »s« folgen!

Befehl	Bedeutung
-f	weibliche Stimme
-m	männliche Stimme
-n	natürliche Betonung
-r	tonlose roboterhafte Aussprache
-p<zahl>	Tonhöhe verändern
-s<zahl>	Sprechgeschwindigkeit verändern

***Tabelle 5.2:** Die Befehle zur Änderung der Sprachausgabe von* Say

Noch ein Tip zu dieser Tabelle: Um von einer männlichen zu einer weiblichen Stimme zu wechseln (oder umgekehrt), reicht »-f« meist nicht aus. Gleichzeitig sollten Sie die Tonhöhe ändern, um eine Frauenstimme zu erhalten. »-f« und »-p250« ergibt zum Beispiel eine realistische Frauenstimme und »-m« und »-p100« eine realistische Männerstimme.

Wenn Sie der merkwürdigen Laute, die Ihr Amiga 500 ausstößt, dann endlich müde sind, können Sie *Say* beenden, indem Sie das Eingabe-Fenster anklicken, nichts eingeben und ein- oder zweimal auf die [Return]-Taste drücken.

5.5.4 NoFastMem

Das Werkzeug *NoFastMem* in der Systemschublade der Workbench-Diskette dient dazu, bestimmte alte Programme auf Ihrem Amiga 500 laufen zu lassen, die zu einer Zeit entwickelt wurden, als es nur den Amiga 1000 mit höchstens 512 Kbyte RAM-Speicher gab. Solche Programme laufen oft auf dem Amiga 500 dann nicht mehr, wenn Sie die Speichererweiterung Amiga 501 eingebaut haben. Damit Sie mit einem solchen Programm arbeiten können, ohne dazu jedesmal die Speichererweiterung ausbauen zu müssen, gibt es *NoFastMem.* Wenn Sie mit einem solchen Programm arbeiten wollen, müssen Sie zunächst *NoFastMem* starten und dann erst das »Oldtimer-Programm«. *NoFastMem* belegt nämlich den gesamten RAM-Speicher, der sich in der Speichererweiterung Amiga 501 befindet (512 Kbyte), und verhindert so, daß das fehlerhafte Programm diesen Speicherbereich benutzen kann. Dieser Speicherbereich wird auch »Fast Memory« (schneller Speicher) genannt, woher *NoFastMem* seinen Namen hat. Wenn Sie später wieder den vollen Speicher nutzen wollen, brauchen Sie *NoFastMem* nur noch einmal starten. Sie können nach dem Start von *NoFastMem* jeweils in der Titelleiste der Workbench verfolgen, wie der freie Speicher ab- und wieder zunimmt.

5.5.5 SlowMemLast

Das Werkzeug *SlowMemLast* ist fast das genaue Gegenteil von *NoFastMem.* Nachdem Sie *SlowMemLast* das erste Mal starten, werden Programme und Daten bevorzugt in den RAM-Bereich oberhalb der ersten 512 Kbyte (Chip-RAM) gelegt. (Ein erneuter Start von *SlowMemLast* hebt diesen Effekt wieder auf.) Der Chip-RAM (der manchmal auch »SlowMem-Bereich« heißt) wird nur dann verwendet, wenn der FastMem-Bereich oberhalb

von 512 Kbyte voll ist oder wenn ein Programm ausdrücklich auf Chip-RAM »besteht«. Das kann dazu führen, daß Programme insgesamt etwas schneller sind, daß Programme, die im FastMem-Bereich liegen, unter gewissen Umständen wirklich etwas schneller laufen. Mehr darüber im letzten Teil dieses Buches.

5.5.6 InitPrinter

Das Werkzeug *InitPrinter* dient zur »Initialisierung« des Druckers, also zur Vorbereitung des Druckens. Es wird nicht unbedingt bei jedem Drucker benötigt. Wenn Sie den von Ihnen verwendeten Drucker aber korrekt mit *Preferences* angemeldet haben und trotzdem nur »Quatsch« aus dem Drucker kommt, sollten Sie *InitPrinter* einmal starten und dann noch einmal versuchen zu drucken.

5.5.7 CLI

Unter Umständen finden Sie im Diskettenfenster oder in der System-Schublade Ihrer Workbench-Diskette noch ein weiteres Werkzeug namens *CLI*. Die Buchstaben »CLI« stehen für *Command Line Interface*. Dahinter verbirgt sich eine zweite, von der Workbench unabhängige Art, den Amiga zu bedienen. Diese Methode arbeitet mit Befehlen, die Sie über die Tastatur eintippen müssen, und Rückmeldungen, die in reiner Textform, ohne Grafiken und Symbol, auf den Bildschirm ausgegeben werden.

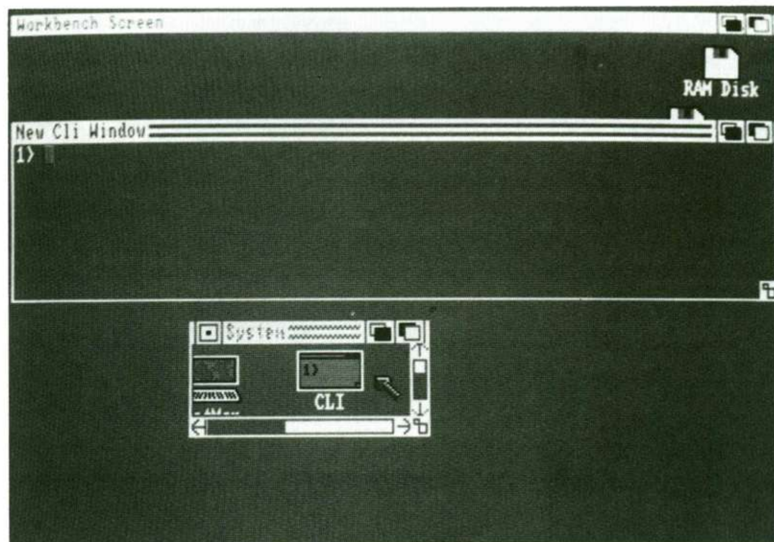


***Bild 5.22:** Das CLI-Piktogramm mit einem CLI-Fenster*

Nähere Erläuterungen zum Thema CLI finden Sie im zweiten Teil dieses Buches (ab Kapitel 7), der sich ausschließlich mit dem CLI beschäftigt. Solange Sie diesen Buchteil nicht durchgearbeitet haben, sollten Sie das Werkzeug CLI nicht starten. Sollte es Ihnen aber doch einmal passieren, daß Sie CLI aus Versehen starten, so warten Sie bitte, bis ein neues Fenster auf dem Bildschirm erscheint. Tippen Sie dann den Befehl *endcli* ein und drücken Sie die [Return]-Taste. Das Fenster müßte daraufhin wieder vom Bildschirm verschwinden (es besitzt kein Schließ-Gadget und kann nur auf die eben beschriebene Weise wieder geschlossen werden).

5.5.8 Format und DiskCopy

In der Systemschublade Ihrer Workbench-Diskette befinden sich neben den bislang beschriebenen noch eine Reihe weiterer Werkzeuge. Auf der Version der Diskette, die diesem Buch zugrunde liegt, waren es zwei Werkzeuge namens *Format* und *DiskCopy*. Bei Ihnen können es aber unter Umständen auch mehr sein. Diese Werkzeuge sind nicht dazu gedacht, von der Workbench aus aufgerufen zu werden. Starten Sie sie deshalb nicht. Falls es Ihnen aus Versehen doch einmal passieren sollte, daß Sie einen Doppelklick auf eines dieser Piktogramme machen, ist das aber auch kein Beinbruch. Diese Programme bewirken dann nämlich entweder überhaupt nichts oder zeigen nur einen Requester, den Sie mit einem Klick auf OK oder CANCEL wieder schließen können.

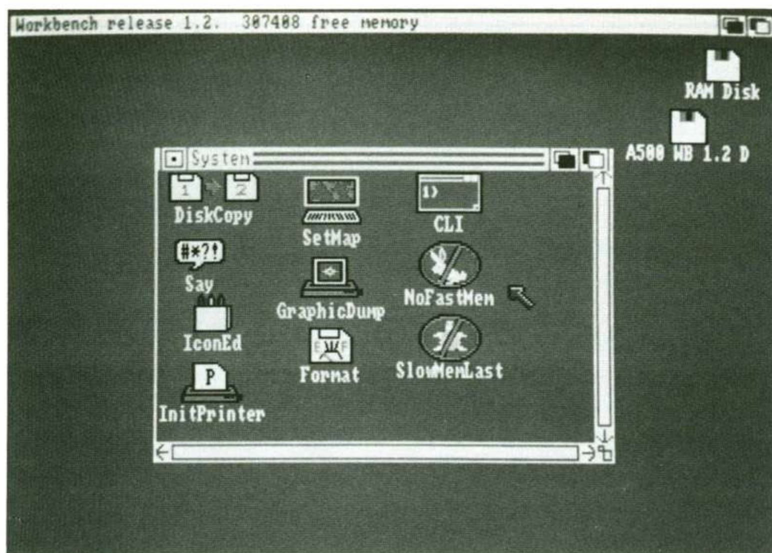


Bild 5.23: Inhalt der System-Schublade

5.6 Werkzeuge auf der Extras-Diskette

Nicht nur auf der Workbench-Diskette, sondern auch auf der Extras-Diskette, die Sie beim Kauf des Amiga 500 ebenfalls erhalten, befinden sich einige Werkzeuge. Neben Amiga-BASIC, das an anderer Stelle noch näher beschrieben wird, sind dies vor allem kleine Hilfsmittel (Tools und Utilities), die für den fortgeschrittenen Anwender gedacht sind und sich in der Schublade *Tools* befinden. Auf eines dieser Hilfsmittel, den Texteditor *MicroEMACS*, wird an anderer Stelle in diesem Buch eingegangen. Dieser und die anderen Werkzeuge der Extras-Diskette, die mir vorlagen, werden aber wahrscheinlich nicht auf der endgültigen Extras-Diskette sein, die Sie erhalten würden, wenn Sie zu dem Zeitpunkt einen Amiga 500 kaufen, da Sie auch dieses Buch lesen. Ich habe deshalb auf Beschreibungen dieser – meist auch sehr speziellen – Werkzeuge verzichtet.

Wenn Sie mehr über die Werkzeuge erfahren wollen, die sich auf Ihrer Extras-Diskette befinden, brauchen Sie nur einen Doppelklick auf das Piktogramm *Hinweise* zu machen, das sich im Diskettenfenster der Extras-Diskette befindet. *Hinweise* ist ein *Notepad*-Projekt, das alle Informationen enthält, die Sie für den Umgang mit den anderen Werkzeugen auf der Extras-Diskette benötigen. *Notepad* wird bei einem Doppelklick auf *Hinweise* deshalb automatisch von der Workbench-Diskette gestartet und die erste Seite des Textes erscheint am Bildschirm. Die weiteren Seiten können Sie betrachten, indem Sie auf das Eselsohr in der unteren linken Ecke des Fensters klicken.

6 Intuition

Wann immer bislang von Eigenschaften der Workbench, wie Menüs und Fenstern, gesprochen wurde, so klang das oft, als wäre die Workbench-Software für alle diese Vorgänge verantwortlich. Dies ist nicht ganz richtig. In Wirklichkeit nutzt die Workbench – und auch die meisten anderen Programme auf dem Amiga 500 – eine Programm-Bibliothek, die sich um alle wichtigen Aspekte der Kommunikation eines Programms mit dem Anwender (also Ihnen) kümmert. Man kann sich diese Programm-Bibliothek als eine Sammlung kleiner Programme vorstellen, die irgendwo im Speicher des Amiga 500 liegt und bei Bedarf von anderen Programmen aufgerufen werden kann. Diese Programm-Bibliothek heißt »Intuition«, was nichts anderes bedeutet, als daß sie es ermöglicht, die Bedienung von Programmen auf dem Amiga 500 möglichst intuitiv zu gestalten.

Intuition ist ein Teil der sogenannten System-Software des Amiga 500, die unter anderem auch die Verwaltung des RAM-Speichers, der Diskettenlaufwerke und des Druckers erledigt. Der Fachmann nennt diesen Teil Benutzerschnittstelle und deutet damit an, daß diese Software »an der Stelle sitzt«, an der der Übergang vom Innenleben des Computers und der Programme darin zur Außenwelt und damit zum Anwender stattfindet.

Intuition steht jedem Programm zur Benutzung zur Verfügung und erfüllt auf diese Art und Weise gleich zwei wichtige Aufgaben. Da sich die Programme nicht alle selbst um sämtliche Details der Programm-Bedienung, wie das Verschieben von Fenstern und die korrekte Anzeige der Maus, zu kümmern brauchen, werden sie kleiner und können schneller geschrieben werden. Zugleich werden die Programme aber auch in gewissen Grenzen standardisiert, das heißt, sie erledigen ähnliche Aufgaben auf ähnliche Weise, da sie gleiche Teile von Intuition dazu verwenden. Für Sie als Anwender hat das vor allem die Konsequenz, daß Sie nicht jedesmal völlig umdenken müssen, wenn Sie ein neues Programm bekommen. In gewissen Aspekten verhalten sich alle Amiga-Programme gleich oder zumindest ähnlich. Ein Paket wie Intuition darf deshalb nicht unterschätzt werden.

Im folgenden werden die wichtigsten Komponenten und Eigenschaften von Intuition beschrieben. Vieles davon werden Sie schon aus den vorangegangenen Kapiteln kennen. Es wird an dieser Stelle aber noch einmal in kompakter Form zusammengetragen. Wenn Sie Schwierigkeiten mit einem der Aspekte haben, die im folgenden beschrieben sind, so können Sie hier immer sehr gut nachschlagen, um »Grundsätzliches« über Fenster, die Maus und ähnliche Dinge zu erfahren, bevor Sie sich dem Handbuch des entsprechenden Programms oder anderen Kapiteln in diesem Buch zuwenden. Sie können dieses Kapitel als eine Art Kurzhandbuch zur Amiga-Software verwenden. Alle grundlegenden Eigenschaften von Amiga-Programmen sind hier zu finden.

Sie sollten die folgenden Abschnitte aber auch dann lesen oder zumindest überfliegen, wenn Sie schon einige Erfahrungen mit dem Amiga 500 haben. Ein paar Dinge werden dort nämlich auch zum ersten Mal – und an keiner anderen Stelle in diesem Buch – erläutert. Das hat einfach damit zu tun, daß manche in die Tiefe gehende Erläuterung in den ersten Kapiteln »das Wesentliche« nur verdeckt beziehungsweise verwirrt hätte. In diesem Kapitel habe ich hingegen versucht, alle Themen so knapp und **vollständig** wie möglich zu behandeln.

Alle folgenden Erläuterungen sind programmunabhängig, das heißt, sie können Ihnen bei der Arbeit mit jedem Programm helfen, da sie beschreiben, wie sich Teile eines »typischen« Amiga-Programms verhalten. Es wird sehr genau getrennt zwischen den allgemeinen Eigenschaften, wie Intuition sie bietet, und den Beispielen in konkreten Programmen, ohne die man natürlich auch nicht auskommen kann. Bei allen Erläuterungen wird aber davon ausgegangen, daß Sie zumindest schon etwas mit der Bedienung des Amiga 500 vertraut sind, wenn vielleicht auch nicht mit allen Details.

Denken Sie jedoch immer daran, daß kein Programm gezwungen wird, Intuition in der beschriebenen Weise zu verwenden. Prinzipiell gibt es immer viele verschiedene Methoden, einen bestimmten Aspekt der Bedienung eines Programms zu lösen. Und es gibt immer auch Programme, die andere Wege gehen, als Intuition sie vorgibt.

6.1 Die Maus

Die Maus ist für die Bedienung des Amiga 500 ein noch wichtigeres Werkzeug als die Tastatur. Sie wird von Intuition (und damit innerhalb aller Programme, die Intuition verwenden) für unzählige Aufgaben eingesetzt. Ohne die Maus sind Sie (fast) verloren – zumindest, wenn Sie dann noch den Amiga 500 bedienen wollen.

6.1.1 Das Funktionsprinzip der Maus

In dem kleinen Kästchen, das »Maus« genannt wird – obwohl die Ähnlichkeit mit dem entsprechenden Tier nicht gerade überwältigend ist – befindet sich eine kleine Kugel. Immer, wenn Sie die Maus auf eine glatte Oberfläche legen und sie verschieben, dreht sich diese Kugel und überträgt diese Drehbewegung auf zwei kleine Rädchen im Innern der Maus. Diese Rädchen senden bei Drehung elektrische Impulse zum Amiga 500, aus denen ein Programm genau schließen kann, in welche Richtung und wie weit Sie die Maus bewegt haben. Intuition

verwendet diese Impulse dazu, einen kleinen Zeiger auf dem Bildschirm zu bewegen, den sogenannten Mauszeiger. Er sieht meist aus wie ein kleiner Pfeil. Sie können ihn aber mit dem Programm *Preferences* nach Ihrem Geschmack abändern. (Der Zeiger ist in Wirklichkeit übrigens ein sogenanntes *Sprite*, über das Sie im letzten Teil des Buches noch mehr erfahren können.)

Die Maus wird richtig gehalten, wenn ihr Schwanz (der Kabelanschluß) nach vorn auf den Bildschirm weist. Bewegt man die Maus dann nach rechts oder links, so bewegt sich auch dieser Zeiger auf dem Bildschirm nach rechts oder links. Bewegt man sie auf der Unterlage nach vorn, geht der Mauszeiger am Bildschirm nach oben. Und zieht man die Maus näher zu sich heran, bewegt sich der Zeiger nach unten.

Wie schnell sich der Mauszeiger über den Bildschirm bewegt, können Sie in gewissen Grenzen selbst bestimmen. Grundsätzlich entspricht die Geschwindigkeit des Zeigers auf dem Bildschirm der Geschwindigkeit, mit der Sie die Maus bewegen. Sie können im Programm *Preferences* oben rechts im Hauptfenster aber auch die prinzipielle Übersetzung zwischen Mausbewegung und Zeigerbewegung wählen. Sie haben die Auswahl zwischen 1, 2 und 4, wobei größere Zahlen eine langsamere, trägere Maus bedeuten. Die Zahlen entsprechen der Anzahl von elektrischen Impulsen, die von einem der beiden Rädchen in der Maus kommen müssen, bevor sich der Zeiger auf dem Bildschirm einen Punkt in die angegebene Richtung bewegt.

6.1.2 Der Mauszeiger

Man verwendet den Mauszeiger normalerweise wie einen zusätzlichen Finger, um auf bestimmte Punkte am Bildschirm zu deuten. Bei präzisen Arbeiten ist es dabei wichtig zu wissen, auf genau welchen Punkt am Bildschirm der Zeiger deutet. Bei dem kleinen Pfeil, der normalerweise innerhalb der Workbench gezeigt wird, liegt dieser »heiße Punkt« an der Pfeilspitze. Ist der Mauszeiger ein kleines Kreuz – wie in einigen Grafik-Programmen – so liegt der heiße Punkt meist genau »auf der Kreuzung«.

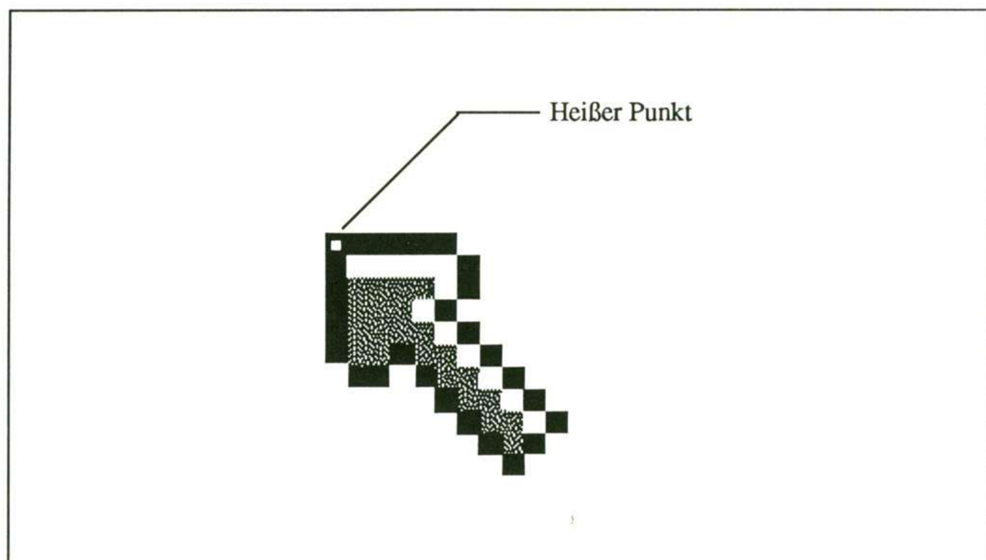


Bild 6.1: Der heiße Punkt eines Mauszeigers

Wenn Sie im Programm *Preferences* (natürlich mit der Maus) auf den Knopf *Edit Pointer* drücken, können Sie den Mauszeiger beliebig Punkt für Punkt ändern. Ihnen stehen dazu drei echte Farben und die »Farbe« Transparent zur Verfügung, die Sie nach Möglichkeit geschickt verwenden sollten, damit der Zeiger auch auf jedem Hintergrund immer gut zu sehen ist.

Sind Sie mit Ihrem Zeiger-Design fertig, können Sie auch noch den heißen Punkt an eine beliebige Stelle setzen, indem Sie auf den Knopf *Set Point* klicken und dann auf den Punkt in der vergrößerten Darstellung des Zeigers klicken, wo sich der heiße Punkt befinden soll. Legen Sie den heißen Punkt aber immer an die Position, die man bei einer bestimmten Zeigerform auch »intuitiv« erwarten würde, bei einem Pfeil zum Beispiel an die Spitze und bei einem Kreuz oder Kreis ins Zentrum. Ansonsten kann der Umgang mit der Maus sehr verwirrend sein. Wenn Sie bei dem Amiga eines Freundes zum Scherz den heißen Punkt des Mauszeigers an das untere Ende des kleinen Pfeils verlegen, so wird er große Schwierigkeiten haben, Fenster auf der Workbench zu öffnen oder zu schließen.

6.1.3 Die schlafende Maus

Der Mauszeiger sieht nicht immer wie ein Pfeil aus. Ein Programm kann diese Form jederzeit ändern. Die Änderung des Mauszeigers ist manchmal eine Mitteilung des Programms. Eine Änderung, die häufig vorkommt, ist der Wechsel vom Pfeil zu einer Sprechblase, in der zwei »Z« erscheinen. Jeder Comic-Leser weiß, was das zu bedeuten hat: jemand schläft oder döst. In diesem Fall ist es die Maus, die schläft. Das heißt, der Amiga 500 ist mit einem

bestimmten Vorgang längere Zeit beschäftigt und kann nicht sofort auf Ihre Eingaben, zum Beispiel einen Mausklick, reagieren.

Andere Programme besitzen noch andere Konventionen für das Aussehen der Maus. Manchmal ändert sich der Mauszeiger, wenn man ihn über bestimmte Gebiete eines Fensters bewegt. Dies bedeutet üblicherweise, daß ein Mausklick in diesem Moment eine bestimmte Wirkung hätte, die durch das Aussehen des Zeigers angedeutet wird. Wählt man im Programm Sonix (einem Kompositions- und Synthesizer-Programm) zum Beispiel eine Note aus und bewegt dann den Mauszeiger über das Notenblatt, so wird der Mauszeiger zur Note. Das bedeutet, daß diese Note auf dem Blatt erscheinen würde, wenn man in diesem Moment klicken würde.

6.1.4 Die Maustasten

Allein das Bewegen des Mauszeigers mit der Maus bewirkt in den meisten Programmen nichts. Falls das Programm nicht ausdrücklich darum bittet, teilt Intuition ihm diese Bewegungen noch nicht einmal mit. Erst wenn einer der Knöpfe auf der Oberseite der Maus niedergedrückt wird, bewirkt dies eine Aktion.

Drückt man die linke Maustaste, die auch oft »Selektionstaste« genannt wird, nieder und läßt sie sofort wieder los, so nennt man dies einen »Mausklick«. Drückt man zweimal schnell hintereinander die Taste nieder und läßt sie wieder los, spricht man von einem »Doppelklick«. Wenn sich ein bestimmtes, von der Umgebung deutlich abgegrenztes Objekt (zum Beispiel ein Piktogramm) unter dem heißen Punkt des Mauszeigers befand, so gilt es durch einen Mausklick üblicherweise als selektiert oder ausgewählt. Ein solcher Mausklick auf ein Piktogramm heißt auch »Auswahl dieses Objekts« und hat üblicherweise noch keine Aktion des Programms zur Folge. Klickt man jedoch auf ein sogenanntes Gadget (siehe unten), so bewirkt ein Mausklick meist eine sofortige Aktion des Programms.

Ein Doppelklick bewirkt bei vielen Objekten (zum Beispiel Piktogrammen), die keine Gadgets sind, eine sofortige Reaktion des Programms. Innerhalb der Workbench und bei einigen anderen Programmen auf dem Amiga 500 ist ein Doppelklick auf ein Objekt der universale »Sesam-öffne-dich«-Befehl. In der Workbench zeigt der Doppelklick zum Beispiel den Inhalt von Disketten und Schubladen an und öffnet (startet) Werkzeuge und Projekte. Viele Programme für den Amiga 500 halten sich an diese Konvention und zeigen ähnliche Reaktionen.

Drückt man die linke Maustaste nieder und bewegt dann die Maus bei festgehaltener Taste, so nennt man dies »Ziehen (mit der Maus)«. Auf diese Weise wird meist ein größerer Bereich auf dem Bildschirm selektiert oder ein Objekt bewegt. Piktogramme auf der Workbench können auf diese Weise zum Beispiel verlegt werden, und in vielen Grafik-Programmen können Sie rechteckige Bereiche markieren, indem Sie in die obere linke Ecke des Bereichs klicken, die Taste dann festhalten und die Maus in die rechte untere Ecke des gewünschten Rechtecks ziehen und dort die Selektionstaste loslassen.

Die rechte Taste auf der Maus heißt Menütaste oder Menüknopf. Drücken Sie diese Taste nieder, erscheinen in der Titelleiste am oberen Rand des Bildschirms die Überschriften (Titel)

der Befehlsmenüs. Auch die Auswahl eines Befehls aus einem dieser Menüs spielt sich weitgehend mit der rechten Maustaste ab. Mehr darüber erfahren Sie weiter unten.

Die rechte Maustaste wird normalerweise ausschließlich für die Menüauswahl verwendet. Klickt man zum Beispiel mit ihr auf ein Objekt oder Gadget, hat dies in der Regel keinerlei Wirkung. Nur die linke Taste bewirkt üblicherweise eine Selektion oder eine Aktion bei Gadgets. Wie bei allen Aspekten von Intuition ist aber kein Programm gezwungen, sich an diese Bedeutung zu halten. Mindestens ein Programm (Deluxe Paint, das vielleicht beste Malprogramm für den Amiga 500) nutzt die Menütaste auch noch für andere Zwecke.

6.1.5 Die »Tastatur-Maus«

Sollten Sie einmal in die Verlegenheit kommen, Ihre Maus nicht gebrauchen zu können oder gebrauchen zu wollen, und müssen trotzdem den Mauszeiger bewegen, können Sie das übrigens auch mit der Tastatur machen. Daß diese Möglichkeit bislang noch nicht erwähnt wurde, liegt einfach daran, daß diese Art der Mausbewegung sehr unpraktisch und mühselig ist.

Zur Bewegung des Mauszeigers benötigen Sie die Commodore-([*C*=]-) links oder die Amiga-([A]-)Taste rechts neben der Leertaste sowie die vier Pfeil-Tasten rechts auf Ihrem Amiga 500. Ich nehme im folgenden einmal an, Sie würden die Commodore-Taste verwenden. Halten Sie einfach die Commodore-Taste fest und drücken auf eine Pfeil-Taste. Das hat den selben Effekt, als hätten Sie die Maus auf ihrer Unterlage in die Richtung bewegt, in die der Pfeil zeigt.

Tippen Sie einmal auf eine Pfeil-Taste, und die Maus bewegt sich einen Punkt in die angezeigte Richtung. Wenn Sie die Pfeil-Taste längere Zeit festhalten, so bewegt sich die Maus mit wachsender Geschwindigkeit in die angegebene Richtung. Wenn Sie den Mauszeiger in größeren Schritten über den Bildschirm springen lassen möchten, müssen Sie gleichzeitig mit der Commodore-Taste und der entsprechenden Pfeil-Taste auch noch eine der beiden Shift-Tasten festhalten. Die dann zustande kommenden Sprünge sind allerdings recht groß – ein Sprung entspricht etwa einer viertel Bildschirmhöhe.

Aber nicht nur die Mausbewegung läßt sich über die Tastatur simulieren. Auch die Maustasten können über die Tastatur simuliert werden. Dazu drücken Sie, nachdem Sie den Mauszeiger zur gewünschten Stelle bewegt haben, gleichzeitig auf die Commodore-Taste und auf eine der beiden [Alt]-Tasten. Dabei entspricht die linke [Alt]-Taste der linken (Selektions-) Taste und die rechte [Alt]-Taste der rechten (Menü-) Taste auf der Maus.

Die Maus auf diese Art und Weise zu simulieren, ist sehr mühselig und verlangt einige Übung. Es ist im Grunde nur dann sinnvoll, wenn Ihre Maus einmal defekt sein sollte und Sie dringende Arbeiten zu erledigen haben, bevor sie repariert werden kann.

6.2 Menüs

Die Menüs, mit denen die meisten Programme des Amiga 500 bedient werden können, sind eine extrem praktische Einrichtung von Intuition, vielleicht der wichtigste Aspekt von Intuition überhaupt. Nahezu alle Programme, mit denen Sie auf dem Amiga 500 arbeiten können, benutzen Menüs. Menüs enthalten alle Befehle, die zu einem bestimmten Zeitpunkt einem Programm erteilt werden können. Sie verhindern, daß man diese Befehle vergißt, da man sie niemals einzugeben, sondern immer nur aus einer Liste (einem Menü) der möglichen Befehle auszusuchen braucht. Gleichzeitig zeigen sie dem Anwender, welche Befehle im Moment möglich sind oder nicht, und welche Einstellung bestimmte »Programm-Schalter« gerade haben.

Alle diese Möglichkeiten bieten die Intuition-Menüs, ohne dafür viel Platz auf dem Bildschirm zu benötigen. Auch auf anderen Computern gibt es schon (andere Formen von) Menüs. Diese sind aber manchmal nur recht umständlich und langsam zu bedienen und nehmen meist sehr viel Platz auf dem Bildschirm ein. Bei den Amiga-Menüs ist das anders. Erst wenn man sie sehen möchte, um einen Befehl daraus auszuwählen, werden die Titel der einzelnen Menüs am oberen Bildschirmrand sichtbar. Und erst wenn man sich für eines der Menüs entschieden hat, werden kurz die einzelnen Menüpunkte (Befehle) sichtbar und verschwinden wieder, sobald man einen davon ausgewählt hat.

6.2.1 Ein Menü

Die Intuition-Menüs sind sogenannte *Pull-down-Menüs* (auf deutsch könnte man das etwa mit *Herunterzieh-Menüs* bezeichnen). Man muß sie nämlich erst aus der Titelleiste (mit der Maus) »herabziehen«, bevor sie sichtbar werden. Erst wenn man die rechte Taste der Maus (die »Menütaste«) drückt, werden die Menütitel sichtbar. Und erst, wenn man – bei gedrückter Menütaste – den Mauszeiger über einen dieser Menütitel bewegt, klappt darunter das eigentliche Menü hervor – meist in Form eines schmalen Streifens, in dem sich mehrere getrennte Felder befinden, die sogenannten »Menüpunkte«.

6.2.2 Menüpunkte

Die einzelnen Punkte, die ein Menü enthält, entsprechen den Befehlen, die man in diesem Menü auswählen kann. Sie können aus einem einzelnen Wort, einem kurzen Satz oder auch einem kleinen Bild bestehen. Enthält ein Menü nur Wörter (Text), spricht man von einem »Textmenü«; enthält es auch Bilder oder Symbole, von einem »grafischen Menü«. Für jeden Menüpunkt ist ein kleines rechteckiges Gebiet im Menü »reserviert«. Diese Gebiete sind manchmal sichtbar durch Linien oder Farbwechsel begrenzt, meist aber auch ohne sichtbare Grenzen problemlos zu unterscheiden.

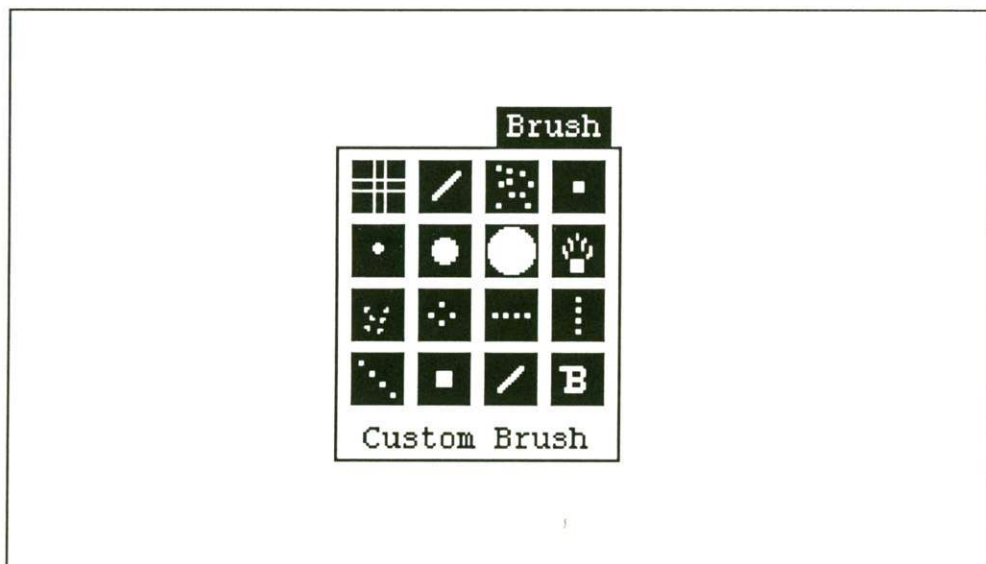


Bild 6.2a: Ein grafisches Menü

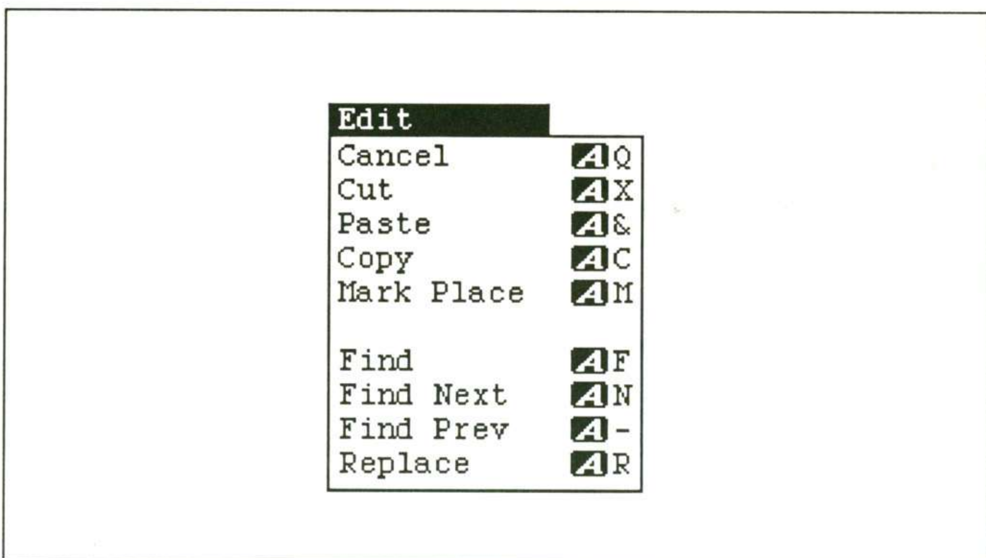


Bild 6.2b: Ein Textmenü

Bewegt man den Mauszeiger (selbstverständlich bei gedrückter Menütaste) in ein solches Gebiet, wird der entsprechende Menüpunkt normalerweise »hervorgehoben«. Es gibt

mehrere Möglichkeiten, einen Menüpunkt hervorzuheben. Bei Menüpunkten, die nur Text enthalten, geschieht das Hervorheben meist durch »Invertieren« der Farben. Das heißt, die Farben, in denen der Text und das umgebende Gebiet am Bildschirm erscheinen, werden ausgetauscht. Es gibt aber auch die Möglichkeit, daß das Gebiet des Menüpunktes zur Hervorhebung plötzlich einen schmalen Rahmen bekommt. Und als dritte Möglichkeit kann das Aussehen des Menüpunktes auch noch vollkommen wechseln, wenn er hervorgehoben werden soll. Dies wird üblicherweise aber nur bei bildhaften Menüpunkten angewendet und auch dort nur selten.

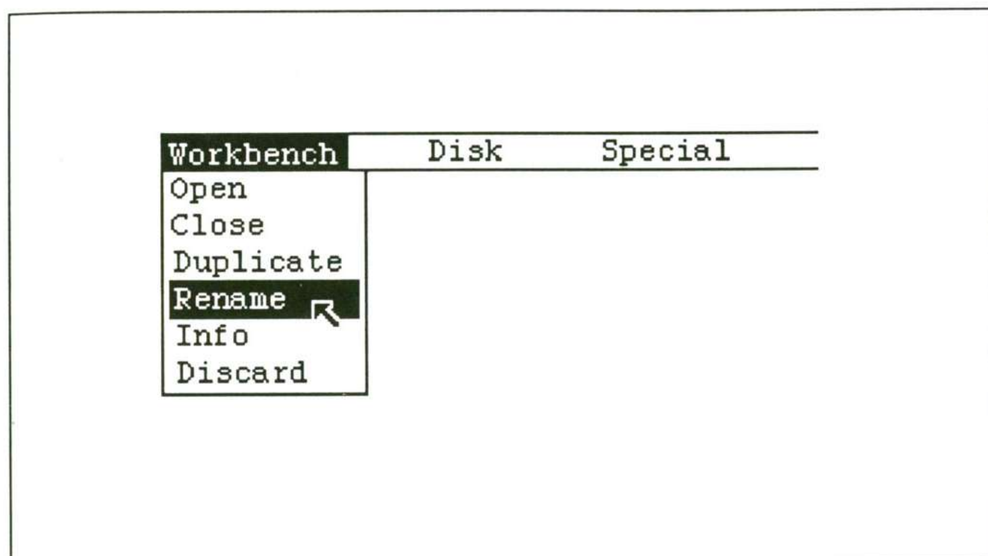


Bild 6.3a: *Hervorhebung eines Menüpunktes durch Invertierung (Farbwechsel)*

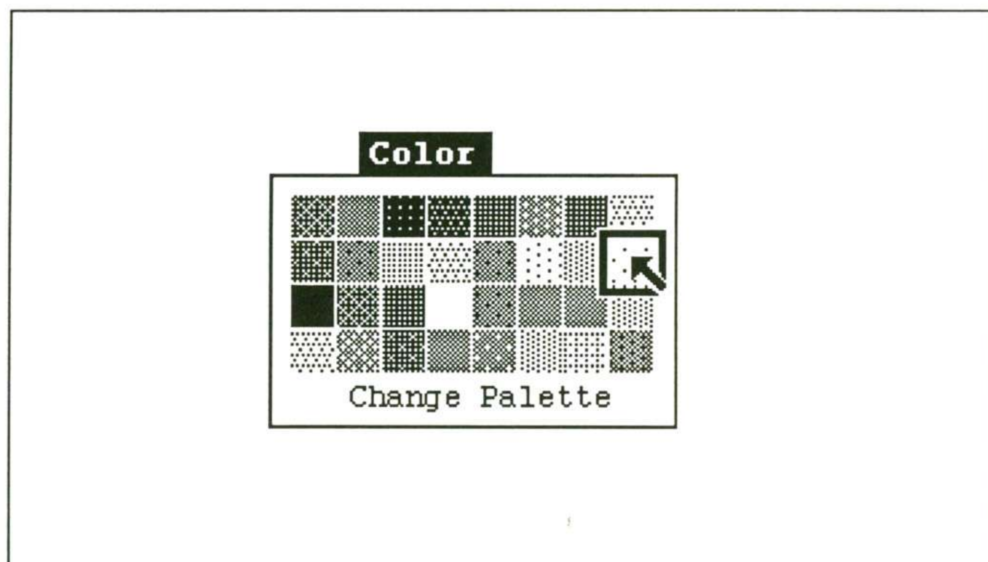


***Bild 6.3b:** Hervorhebung eines Menüpunktes durch Umrahmung*

In jedem Menü ist immer nur ein Punkt hervorgehoben. Bewegt man den Mauszeiger über das Gebiet eines anderen Punktes, so wird dieser hervorgehoben und die Hervorhebung des letzten Punktes verschwindet. Läßt man die Menütaste los, während ein Menüpunkt hevorgehoben ist, so gilt das als Auswahl dieses Punktes aus dem Menü. Eine solche Menüauswahl führt üblicherweise zu einer Aktion des Programms. Sie entspricht also dem, was man auf anderen Computern einen »Befehl an das Programm« nennen würde.

Haben Sie es sich nach dem Herunterklappen eines Menüs anders überlegt und wollen keinen Befehl daraus auswählen, bewegen Sie den Mauszeiger einfach ganz aus dem Rechteck, das das Menü begrenzt, heraus und lassen dann die Menütaste los!

Es gibt noch eine zweite Möglichkeit der Menüauswahl, die besonders dann praktisch ist, wenn Sie nacheinander mehrere Befehle aus einem oder mehreren Menüs auswählen wollen. Sie wurde bisher noch nie erwähnt (und funktioniert in einigen wenigen Programmen nicht)! Hierzu müssen Sie auf die (linke) Selektionstaste der Maus drücken, während sich der Mauszeiger über dem gewünschten Punkt befindet (bei gedrückter Menütaste also). Diese Möglichkeit ist besonders dann praktisch, wenn man schnell hintereinander mehrere Menü-Auswahlen treffen will. Im *Notepad* kann man so zum Beispiel in einem schnellen Arbeitsgang den Text kursiv, fett und unterstrichen gestalten und braucht dazu nicht dreimal hintereinander die Menüs aufzurufen.

Nicht alle Menüpunkte sind zu jedem Zeitpunkt wählbar. Manchmal ist ein bestimmter Befehl in einer bestimmten Situation nicht sinnvoll. Intuition gibt dem Programm die Möglichkeit, dies dem Benutzer auch kundzutun. Hierzu können einzelne Punkte eines Menüs deaktiviert

werden. Deaktivierte Menüpunkte wirken üblicherweise gepünktelt und verschwommen. Bei Textmenüs spricht man auch von »Geisterschrift«. Kommt der Mauszeiger über einen auf diese Art deaktivierten Menüpunkt, so wird er auch nicht hervorgehoben, um Ihnen ganz deutlich zu zeigen, daß er nicht ausgewählt werden kann. Nur wenn ein Menüpunkt hervorgehoben ist, während die Menütaste gelöst wird oder die Selektionstaste gedrückt wird, gilt er als ausgewählt!

6.2.2 Submenüs

Nicht jeder Menüpunkt kann »selbst« ausgewählt werden oder entspricht einem Befehl an das Programm. Manche Menüpunkte sind nämlich in Wirklichkeit gar keine echten Menüpunkte, sondern der Titel eines anderen Menüs. Sobald ein solcher Menüpunkt hervorgehoben wird, erscheint (meistens rechts daneben) ein neues kleines Menü. Solche Menüs heißen »Submenüs« (Untermenüs), und die normalen Menüs, die Sie bisher kennengelernt haben, heißen in diesem Zusammenhang »Hauptmenüs«. Einen Menüpunkt, der ein Submenü »hervorbringt«, nennt man manchmal auch »Submenü-Titel«.

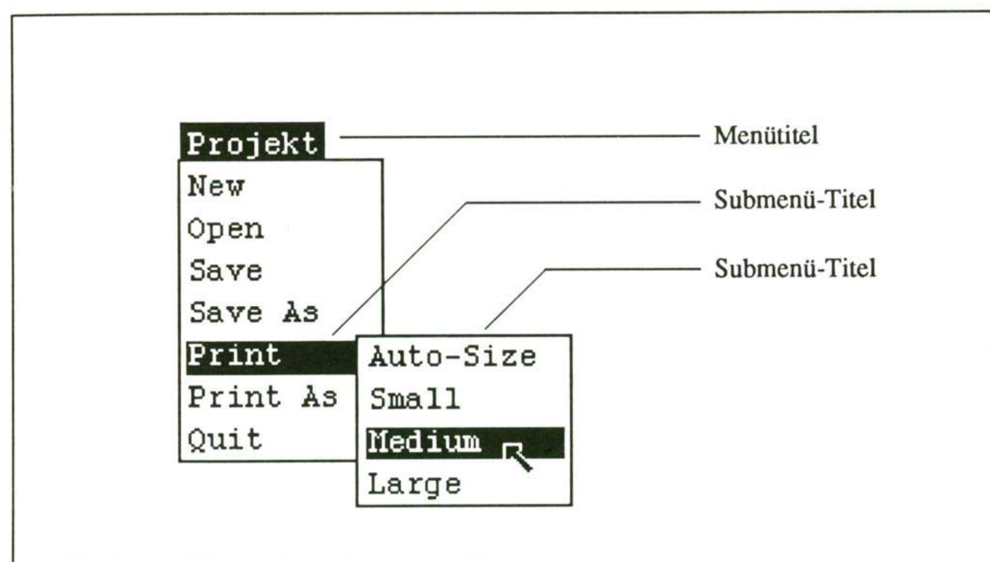


Bild 6.4: Ein Submenü

Sie müssen nun den Mauszeiger in das Gebiet des Submenüs bewegen und dort eine Auswahl treffen. Die Auswahl des Submenü-Titels im Hauptmenü bewirkt alleine nichts.

Ein Tip zu Submenüs: Wenn Sie einen Punkt auswählen wollen, der in einem Submenü ziemlich weit unten liegt, so unterliegt man manchmal der Versuchung, ihn direkt mit der Maus anzufahren, also ohne zunächst nach rechts an den Anfang des Submenüs zu gehen, gleich

schräg nach unten zu fahren. Tun Sie das nicht! Dabei verschwindet das Submenü nämlich meistens wieder, weil man unterdessen einerseits das Gebiet des Submenü-Titels verlassen hat, andererseits aber auch nicht über dem Gebiet des Submenüs ist. Gewöhnen Sie sich deshalb an, bei Submenüs zunächst immer mit der Maus nach rechts zu fahren und den obersten Punkt darin anzusteuern, bevor Sie andere Punkte anwählen!

6.2.3 Markierte Menüpunkte

Manche Punkte in einem Menü entsprechen »Schaltern« des Programms, also dauerhaften Einstellungen, die einen oder mehrere Werte haben können. Die meisten Schalter enthalten nur zwei mögliche Positionen: An und Aus. Das Programm *Clock* kann hierzu als Beispiel dienen. *Clock* (die Amiga-Uhr) zeigt die Uhrzeit nämlich entweder digital (mit Zahlen) oder analog (mit Zeigern) an. Man könnte auch sagen, »der Schalter Digital steht auf An oder Aus«.

Um zu zeigen, welche Stellung ein Schalter hat, werden die Schalter, die »An« sind, meist mit einem Häkchen versehen beziehungsweise »abgehakt«. Wenn Schalter sich wechselseitig ausschließen – wie zum Beispiel *Analog* und *Digital* – so verschwindet das Häkchen bei einem (oder mehreren) Menüpunkte(n), sobald es bei einem anderen auftaucht. (Das *Style*-Menü des Werkzeugs *Notepad* bietet ein schönes Beispiel für dieses Verhalten.)

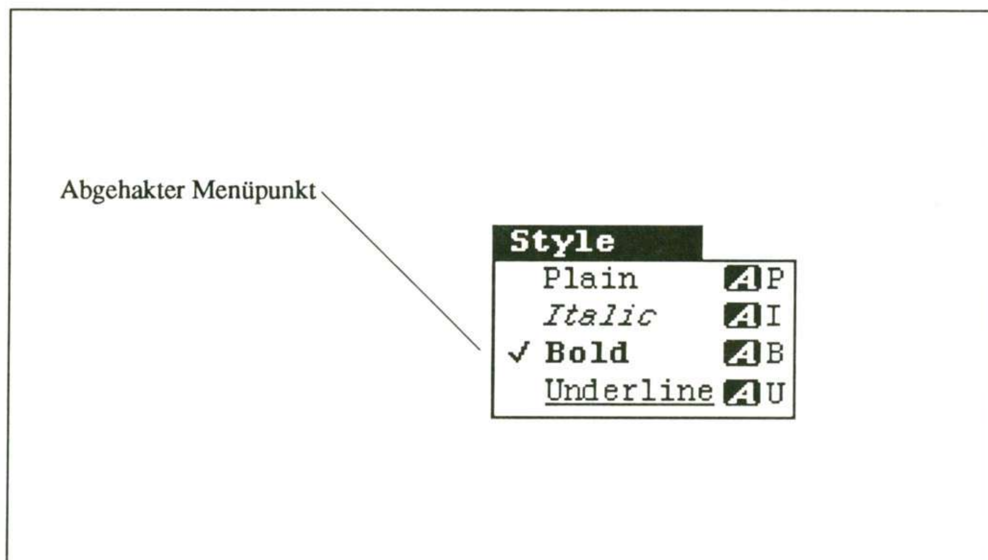


Bild 6.5: *Ein abgehakter Menüpunkt*

Die Auswahl eines solchen Menüschalters kann verschiedene Bedeutungen haben. Wie der Schalter interpretiert wird, ist Sache des zuständigen Anwendungsprogramms. Manchmal legt eine Menüauswahl einen Schalter um, das heißt, wenn er an war, geht er aus und umgekehrt.

Manchmal aber bedeutet die Auswahl stets das Einschalten und zum Ausschalten muß dann ein anderer Menübefehl ausgewählt werden. Die meisten Menüschalter, die Sie bislang kennengelernt haben, gehören dieser zweiten Kategorie an.

6.2.4 Tastaturäquivalente

Menüs sind zwar recht praktisch und vor allem sehr leicht zu erlernen. Es kann aber auch sehr lästig werden, die Maus in die Hand zu nehmen und einen Menüpunkt auszuwählen, wenn man gerade beide Hände auf der Tastatur hat. Die Textverarbeitung ist ein typischer Anwendungsfall, in dem Menüs für bestimmte häufig anfallende Aufgaben auf Dauer einfach zu langsam und umständlich sind.

Deshalb sieht Intuition die Möglichkeit vor, einen Menüpunkt auch mit der Tastatur auswählen zu können. Hierzu muß man die Amiga- oder die Commodore-Taste festhalten und dann einen Buchstaben tippen. Dies hat dann dieselbe Wirkung, als hätte man den entsprechenden Menüpunkt mit der Maus ausgewählt und heißt Tastaturäquivalent eines Menüpunktes, weil eine Tastenkombination mit einem Menübefehl gleichbedeutend (»äquivalent«) ist.

Welcher Menüpunkt mit welcher Tastenkombination äquivalent ist, wird im Menü selbst angezeigt. Neben dem eigentlichen Menüpunkt erscheint dann das Symbol der Amiga-Taste und der entsprechende Buchstabe.

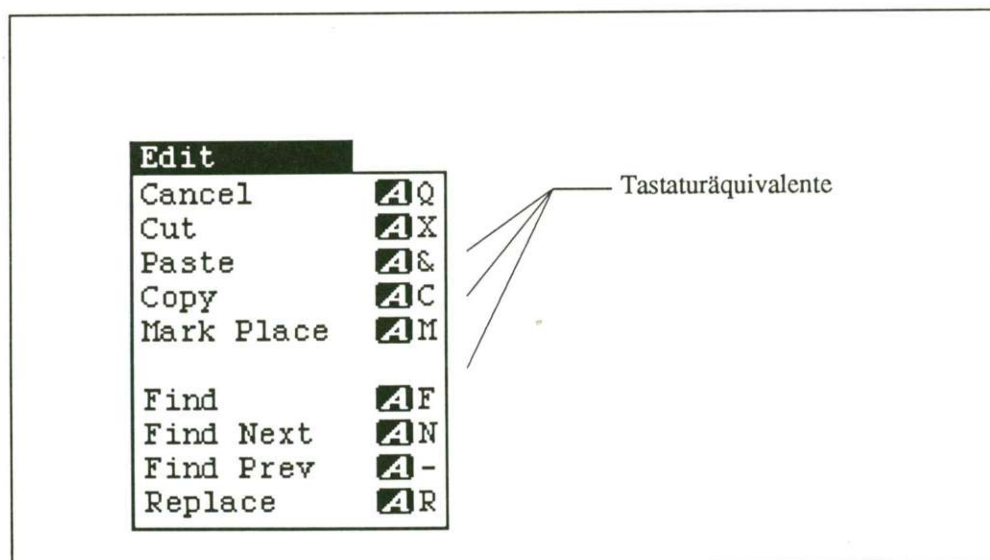


Bild 6.6: Menüpunkte mit Tastaturäquivalent

6.3 Fenster

Neben den Menüs dürften die Fenster wohl das wichtigste Konzept der Amiga-Software sein. Der geschickte Umgang mit Fenstern, zum Beispiel die übersichtliche Positionierung vieler Fenster auf dem Bildschirm, trägt sehr viel zum zügigen Arbeiten mit dem Amiga 500 bei.

6.3.1 Die Philosophie der Fenster

Die Idee, die hinter Fenstern steckt, ist das Arbeiten mit Blättern auf einem Schreibtisch. (Man hört in diesem Zusammenhang deshalb oft auch den Begriff *desktop metaphor*; zu deutsch *Schreibtisch-Metapher*). Jedes Fenster entspricht einem Blatt Papier und kann hin- und hergeschoben, im Stapel der überlappenden Blätter nach unten gelegt oder nach oben geholt und am Ende auch ganz beiseite gelegt werden.

Zu dieser Idee, die noch nichts mit dem Namen »Fenster« zu tun hat, kommt als zusätzliches Konzept noch dazu, daß Fenster in ihrer Größe verändert werden können und vor allem, daß sie die Fähigkeit besitzen, einen Ausschnitt aus einem eventuell viel größeren Bild in ihrem Innern zeigen können. Sie sind also Fenster, mit denen man in ein viel größeres Bild hinausschaut.

Wenn ein Intuition-Fenster einen solchen Ausschnitt aus einem größeren Bild zeigt, besitzt es meist auch Vorrichtungen zum Verschieben des sichtbaren Ausschnitts. Diese Vorrichtungen sind die sogenannten »Rollbalken«, auf die später in diesem Kapitel noch näher eingegangen wird.

6.3.2 Ein typisches Fenster

Ein typisches Intuition-Fenster besteht aus zwei großen Bereichen: dem Fenster-Rahmen und dem Fenster-Inhalt. Intuition kümmert sich fast nur um den Fenster-Rahmen, während für das, was im Innern des Fensters (im Inhalt) sichtbar ist, im wesentlichen das für dieses Fenster zuständige Anwendungsprogramm verantwortlich ist.

Der wichtigste Teil des Fenster-Rahmens ist die Titelleiste, in der unter anderem auch der Name des Fensters erscheint. Sie können die meisten Fenster beliebig auf dem Bildschirm verschieben, indem Sie mit der Maus diesen Namen oder die daneben sichtbare gestreifte Fläche ergreifen. Während Sie das Fenster bewegen, folgt ein flimmerndes Rechteck Ihren Bewegungen. Dieses Rechteck symbolisiert die Fläche, die das Fenster einnehmen würde, wenn die Maustaste in diesem Augenblick gelöst würde.

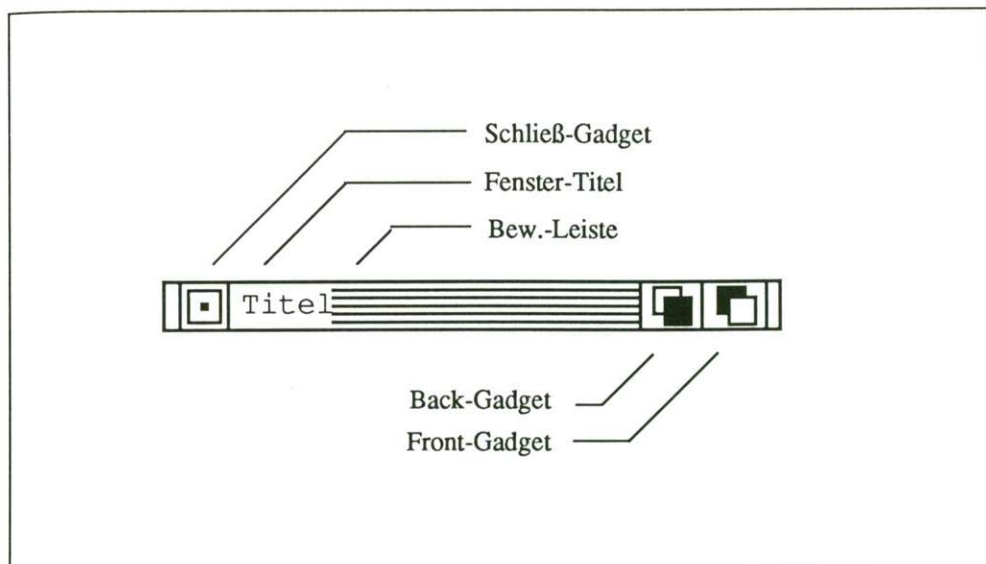


Bild 6.7: Die Titelleiste eines Fensters

Achten Sie beim Verschieben eines Fensters aber darauf, daß Sie wirklich den Namen oder die gestreifte Fläche der Titelleiste »ergreifen«. Die anderen Gebiete der Titelleiste haben andere Funktionen, die Sie nicht versehentlich betätigen sollten.

Links neben dem Namen besitzen manche Fenster ein kleines Kästchen. Dies ist das »Schließ-Gadget«, mit dem das Fenster geschlossen bzw. vom Bildschirm entfernt werden kann. (Mehr über Gadgets erfahren Sie noch später in diesem Kapitel). Wenn Sie (mit der linken Maustaste) in dieses Kästchen klicken und die Maustaste wieder loslassen, solange der heiße Punkt des Zeigers innerhalb des Kästchens ist, so wird das Fenster geschlossen.

Nicht alle Fenster besitzen ein Schließ-Gadget. Fenster, die kein solches Hilfsmittel haben, können entweder gar nicht oder nur durch einen Menübefehl geschlossen werden. Eine dritte Möglichkeit findet man bei den CLI-Fenstern, über die in den Kapiteln, die sich mit dem CLI beschäftigen, mehr zu erfahren ist. Diese Fenster können zwar weder mit einem Schließgadget noch mit einem Menübefehl, aber auf andere Weise doch wieder geschlossen werden.

Rechts neben dem Namen des Fensters und der gestreiften Fläche liegen oft zwei weitere Gadgets: das »Back-« und das »Front-Gadget«. Diese beiden Gadgets dienen zur Manipulation der Reihenfolge der Fenster im überlappenden Stapel von Fenstern. Wenn sich Fenster überlappen, so kann ja nur eins davon (das oberste) komplett sichtbar sein. Alle anderen werden teilweise oder ganz verdeckt. Betätigt man das Front-Gadget eines Fensters (durch einfaches Anklicken), so wird dieses Fenster zum obersten im Stapel. Betätigt man das Back-Gadget, so wird das Fenster zum untersten im Stapel, was oft sogar dazu führt, daß es überhaupt nicht mehr sichtbar ist.

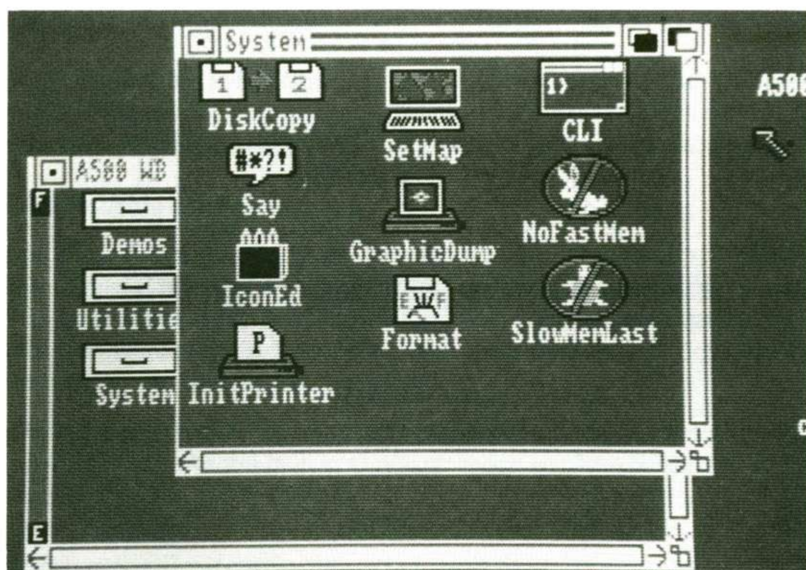


Bild 6.8a: Vor der Betätigung des Front-Gadgets eines Fensters

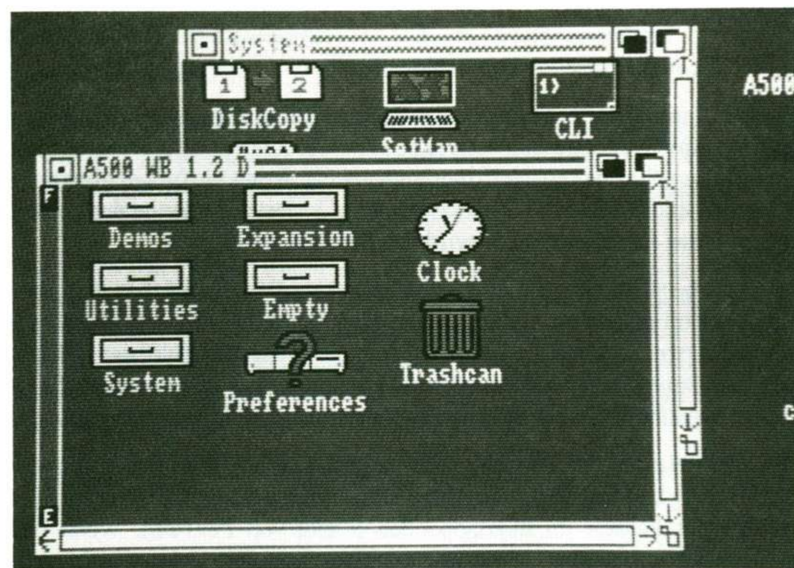


Bild 6.8b: Nach der Betätigung des Front-Gadgets eines Fensters

Unten rechts in einem Fenster befindet sich meist noch ein weiteres Gadget: das »Größen-Gadget«. Dieses Gadget wirkt nicht, wie die Gadgets der Titelleiste, als eine Art Schalter, sondern kann mit der Maus ergriffen und bewegt werden. Die Stelle, an die man es hinlegt (indem man die Maustaste wieder losläßt), wird danach zur neuen unteren linken Ecke des entsprechenden Fensters.

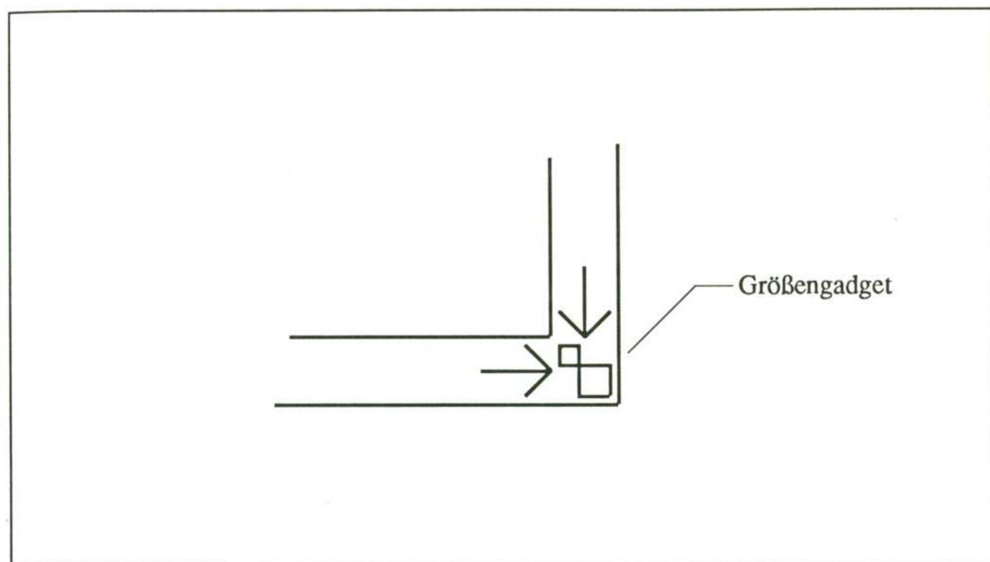


Bild 6.9: Das Größen-Gadget eines Fensters (vergrößert)

Bewegt man das Größen-Gadget, so wird die Fläche, die das Fenster einnehmen würde, wenn die Maustaste in diesem Augenblick gelöst würde, ähnlich wie beim Bewegen eines Fensters permanent durch eine flimmernde Umrißlinie angezeigt. Bei vielen Fenstern hat die Fläche eine maximale und minimale Ausdehnung, die nicht über- oder unterschritten werden kann. Erreicht man bei der Mausbewegung diese Grenzen, »stößt die Maus an« und kann den Mauszeiger – und damit den flimmernden Rahmen – nicht mehr weiter in die angegebene Richtung bewegen.

6.3.3 Rollbalken

Viele Intuition-Fenster sind Öffnungen (Fenster), durch die man in größere Abbildungen und Texte »hineinsehen« kann und die einen Ausschnitt dieser größeren Fläche zeigen. Wie die Größe des Ausschnitts geändert werden kann (mit dem Größen-Gadget), wurde im letzten Abschnitt erläutert. Mit Hilfe der Rollbalken, die manche Fenster am rechten und am unteren Rand enthalten, können Sie den sichtbaren Ausschnitt bewegen. Mit einem vertikalen Rollbalken am rechten Fensterrand kann man den Ausschnitt auf und ab bewegen, mit einem

horizontalen Rollbalken nach links und nach rechts. Manche Fenster besitzen keine Rollbalken, manche nur einen horizontalen oder vertikalen und manche beides. Dies hängt hauptsächlich von der Art der Darstellung im Fensterinnern und ihrer horizontalen und vertikalen Ausdehnung ab. Viele Fenster, die keine Rollbalken besitzen, haben gleichzeitig auch eine unveränderliche Größe (kein Größen-Gadget).

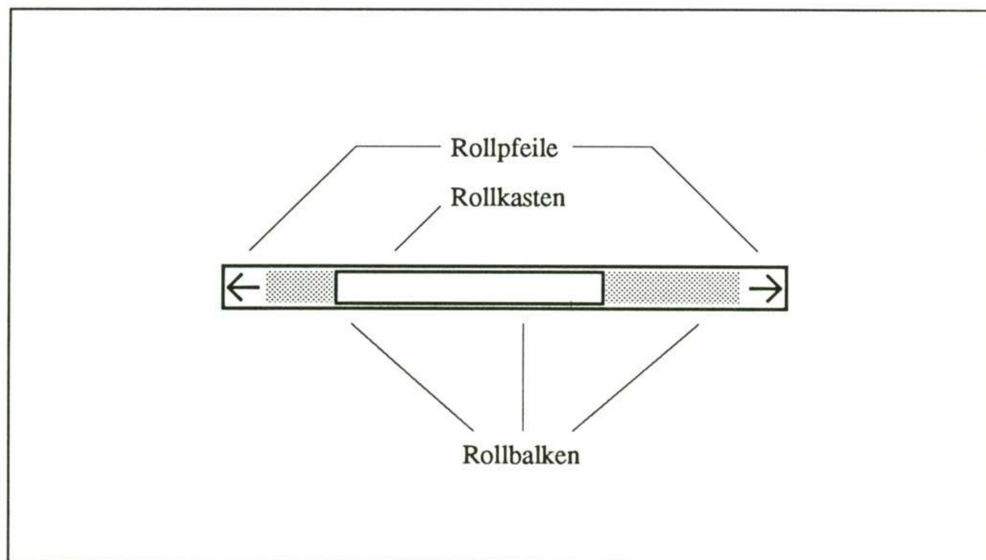


***Bild 6.10:** Ein Rollbalken*

Ein Rollbalken besteht im wesentlichen aus drei Komponenten (Gadgets): zwei »Rollpfeilen« und einem »Rollkasten«. Die Pfeile an beiden Enden eines Rollbalkens verschieben, wenn sie betätigt (angeklickt) werden, den Bildausschnitt um ein gewisses Stück in die angegebene Richtung. Die Größe dieses »Schritts« oder »Sprungs« ist Sache des Programms – meist ist es die Hälfte oder ein Drittel der Fensterhöhe beziehungsweise -breite. Wenn Sie sich in größeren Stücken in eine Richtung bewegen wollen, so klicken Sie in die blaue Fläche zwischen dem Rollpfeil, der in die gewünschte Richtung weist, und dem Rollkasten. Der Ausschnitt springt dann ungefähr eine volle Fensterhöhe beziehungsweise -breite in die entsprechende Richtung.

Der Rollkasten symbolisiert den dargestellten Ausschnitt (in horizontaler beziehungsweise vertikaler Richtung) der ganzen Fläche, der im Fensterinnern gezeigt wird. Der blaue Hintergrund unter dem Rollkasten symbolisiert das gesamte Bild. Je größer der gezeigte Teil des gesamten Bildes ist, desto größer ist auch der Rollkasten. (Wenn das Fenster die ganze Fläche zeigt, nimmt der Rollkasten den gesamten Platz zwischen den beiden Pfeilen ein.) Wenn die Rollpfeile betätigt werden, bewegt sich auch der Rollkasten in die angegebene

Richtung. Und die Breite der blauen Streifen links und rechts vom Rollkasten deutet immer an, wie groß die Teile des Bildes sind, die sich in dieser Richtung noch verbergen.

Der Rollkasten ist aber nicht nur eine Anzeige. Er kann zugleich auch selbst für das Verschieben des Bildausschnittes eingesetzt werden. Hierzu kann man ihn mit der Maus »ergreifen« und ihn dann zu der gewünschten Position schieben. Entweder schon während dieser Verschiebung oder spätestens dann, wenn Sie die Maustaste wieder loslassen, »springt« dann auch der Bildausschnitt im Fenster zu der markierten Stelle.

6.3.4 Fenster und Werkzeuge

Fenster können zwei Zustände einnehmen, die durch ein leicht unterschiedliches Aussehen der Titelleiste angedeutet werden. Ist der Name des Fensters gut lesbar und die Fläche rechts daneben liniert, so ist das Fenster »aktiv«. Sind der Name und die Fläche rechts daneben jedoch gepünktelt, so ist das Fenster »inaktiv«. Ob ein Fenster aktiv oder inaktiv ist, hat vor allem im Zusammenhang mit Werkzeugen große Bedeutung – stammen alle Fenster auf dem Bildschirm von einem einzigen Werkzeug oder der Workbench, spielt es keine so große Rolle.

Intuition speichert nämlich zu jedem Fenster, von welchem Werkzeug es geöffnet wurde. (Ein Werkzeug kann ein oder mehrere Fenster öffnen.) Klickt man mit der Maus auf ein Fenster, so wird es zum aktiven Fenster. (Im Gegensatz zu Fenster-Systemen auf anderen Computern muß das aktive Fenster nicht unbedingt mit dem obersten Fenster im Stapel identisch sein.) Gleichzeitig wird aber auch das Werkzeug, welches das Fenster geöffnet hat, zum aktiven Programm – sofern es das nicht schon war. Dieses Werkzeug »übernimmt« dadurch zum Beispiel die Titelleiste des Bildschirms, und wenn man nun die Menütaste auf der Maus niederdrückt, erscheinen die Menüs des entsprechenden Werkzeugs.

6.4 Gadgets

In diesem Kapitel war schon mehrfach von Gadgets die Rede. Gadgets sind sehr »elementare« Komponenten von Intuition, die für eine Vielzahl von Aufgaben eingesetzt werden können, die sich aber auch alle »irgendwie« ähneln. Gadget (sprich »Gädsched«) heißt übersetzt soviel wie *Schalter*, *Gerät* oder *Bedienungselement*, und genau darum handelt es sich auch. Es sind meist kleine Objekte, die vor allem auf Mausklicks und Mausbewegungen reagieren, wenn sich der Mauszeiger über ihnen befindet.

Zu jedem Gadget gehört ein genau abgegrenztes Gebiet auf dem Bildschirm. Befindet sich der heiße Punkt des Mauszeigers über diesem Gebiet und man drückt die linke Maustaste, so wird das Gadget aktiv. Ähnlich einem Menüpunkt wird dies meist auch durch eine Hervorhebung des Gadgets deutlich signalisiert. Diese Hervorhebung besteht meistens aus einem Farbwechsel der einzelnen Komponenten des Gadgets. Es kann aber auch durch einen kompletten Austausch des momentanen Aussehens gegen ein völlig neues Bild hervorgehoben werden. Letzteres ist allerdings etwas unüblich, da ein solches Verhalten leicht verwirrend wirkt.

Das Niederdrücken der Maustaste betätigt aber nicht unbedingt ein Gadget. Das heißt, es führt nicht unbedingt zu einer Reaktion des zuständigen Programms. Einige Gadgets werden erst dann betätigt, wenn man die Maustaste wieder losläßt, während sich der heiße Punkt des Mauszeigers (immer noch oder schon wieder) innerhalb des Gadget-Gebiets befindet. Andere Gadgets reagieren bereits auf das Drücken der Maustaste. Wiederum andere Gadgets bewirken nur dann etwas, wenn man die Maus bei gedrückter Selektionstaste und damit dann das Gadget »mitnimmt«.

Dies läßt sich durchaus mit Bedienungselementen vergleichen, die uns aus dem täglichen Leben bekannt sind. Manche Tipptasten oder Sensoren an elektrischen Geräten reagieren schon auf Berührung. Andere müssen niedergedrückt und wieder losgelassen werden. Bei Schiebe- und Drehreglern hingegen bewirkt nur eine Bewegung dieses »Gadgets« etwas. Kombinationen der unterschiedlichen Reaktionsmuster gibt es bei Intuition-Gadgets genauso wie in der Wirklichkeit.

6.4.1 Tasten- und Sensoren-Gadgets

Einige Gadgets, die sich wie Tasten oder Sensoren an elektrischen Geräten verhalten, haben Sie ja bereits kennengelernt. Die Rollpfeile in Rollbalken, die Schließ-, Front- und Back-Gadgets in den Titelleisten von Fenstern verhalten sich wie solche Schalter. Bei allen diesen Gadgets gilt aber, daß sie erst dann als betätigt gelten, wenn die Maustaste nach dem Klick in das Gebiet des Gadgets auch wieder gelöst wird, während sich der Mauszeiger immer noch in diesem Gebiet befindet. Wenn Sie aus Versehen ein solches Gadget angeklickt haben, gibt es also immer noch eine Chance, »mit heiler Haut davonzukommen«. Bewegen Sie einfach den Mauszeiger zur Seite, aus dem Gebiet des Gadgets hinaus, und lassen Sie erst dann die Taste los.

Weiterhin gehören in die Kategorie der Gadgets natürlich auch als sozusagen reinstes Beispiel die echten »Knöpfe« oder »Tasten«, die manchmal innerhalb von Fenstern auftauchen. SAVE und QUIT, die am unteren Rand des Fensters erscheinen, wenn der Befehl *Info* auf der Workbench aufgerufen wird, sind zwei schöne Beispiele für solche mit der Maus zu betätigenden Knöpfe.

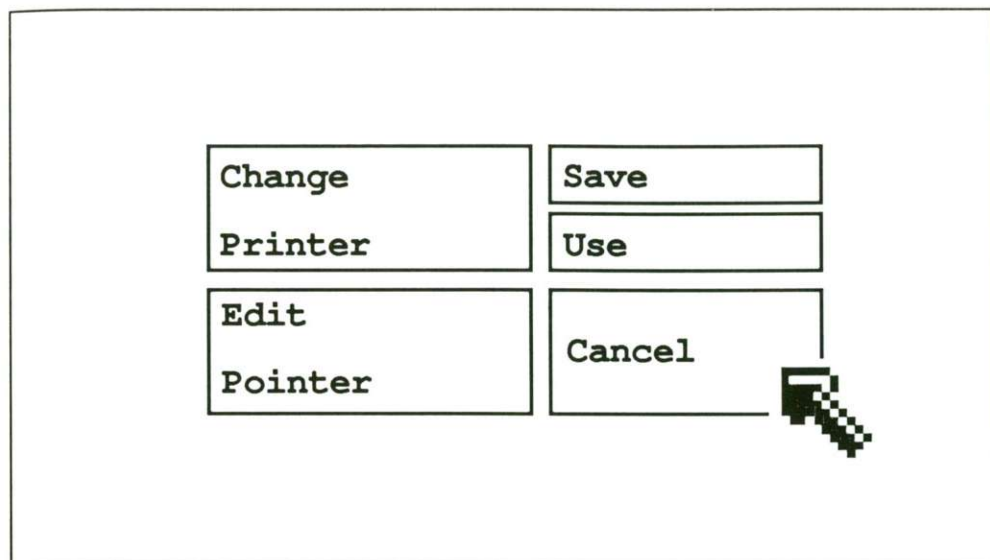


Bild 6.11: Tasten-Gadgets aus dem Programm Preferences

Aber auch die blauen Flächen zwischen dem Rollkasten und einem Rollpfeil oder links und rechts neben einem der Schieberegler in den Preferences sind eine Art Gadgets. Klickt man zum Beispiel in einem Rollbalken in die Fläche zwischen Rollkasten und Rollpfeil, so springt der Rollkasten ein großes Stück in die angegebene Richtung. Anders als bei den bisher kennengelernten Beispielen reagieren diese Flächen schon auf das Niederdrücken und nicht erst auf das Loslassen der Selektionstaste. Man könnte sie deshalb in die Unterkategorie Sensoren einordnen, da sie bereits bei »Berührung« betätigt werden.

6.4.2 Schiebe- und Drehregler

Die zweite wichtige Kategorie von Gadgets sind die Schiebe- und Drehregler. Schieberegler haben wir bei den bisher vorgestellten Programmen schon eine ganze Menge gesehen, selbst wenn sie nicht immer diesen Namen trugen. Drehregler sind theoretisch auch möglich, werden aber bisher von keinem existierenden Programm verwendet – wahrscheinlich, weil sie sich nicht so gut für eine Bedienung mit der Maus eignen.

Schieberegler »in Reinkultur« werden zum Beispiel sehr häufig für die Farbauswahl verwendet. (Aber auch die Rollkästen in Rollbalken sind eine Art Schieberegler.) Im Hauptfenster des Werkzeugs Preferences tauchen zum Beispiel solche Regler auf. Das folgende Bild stammt allerdings aus dem Grafikprogramm DeluxePaint II.

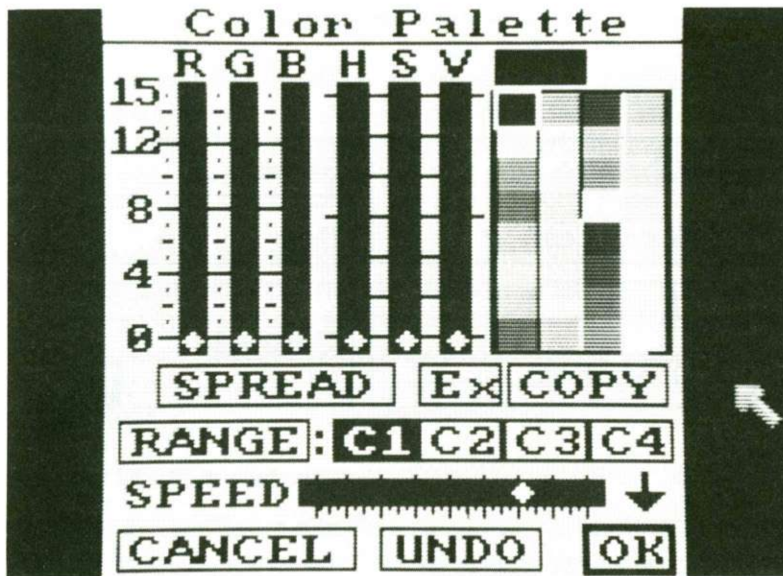


Bild 6.12: Sechs Schieberegler

Schieberegler werden immer betätigt, indem man den eigentlichen Knopf oder Griff des Reglers anklickt, die Selektionstaste dann aber festhält und den Mauszeiger in die gewünschte Richtung verschiebt. Der Knopf folgt dabei den Bewegungen des Mauszeigers und bleibt an der letzten Position liegen, wenn die Maustaste gelöst wird. Fast alle Schieberegler begrenzen die Richtung, in die der Knopf verschoben werden kann, auf die Horizontale oder Vertikale. Selbst wenn der Mauszeiger das Gebiet des Gadgets »in der falschen Richtung« verläßt, bleibt der Knopf auf einen schmalen Streifen begrenzt. Eine Ausnahme bildet in dieser Hinsicht der »Regler«, mit dem die Bildschirmzentrierung geändert werden kann (in der Mitte des Hauptfensters von *Preferences*). Er kann innerhalb seines einschränkenden Rahmens beliebig in alle Richtungen verschoben werden und gehört damit auch in die Kategorie der Schieberegler.

Ebenfalls in diese Kategorie könnte das Größen-Gadget eines Fensters eingeordnet werden, obwohl die Analogie zu Reglern an elektrischen Geräten in der Wirklichkeit hier wohl nicht mehr funktioniert. Aber auch das Größen-Gadget dient als ergriffen, sobald man darüber mit der linken Maustaste klickt. Es folgt dann jeder Mausbewegung, solange die Selektionstaste gedrückt bleibt.

6.4.3 Text-Gadgets

Fast immer, wenn ein kurzer Text eingegeben werden muß, bedienen sich die Amiga-Programme dazu eines Text-Gadgets. Text-Gadgets sind auch immer in den Info-Fenstern der Workbench enthalten. Sie werden zum Beispiel dazu verwendet, um einen Kommentar zu einem bestimmten Werkzeug oder einer Schublade zu vermerken.

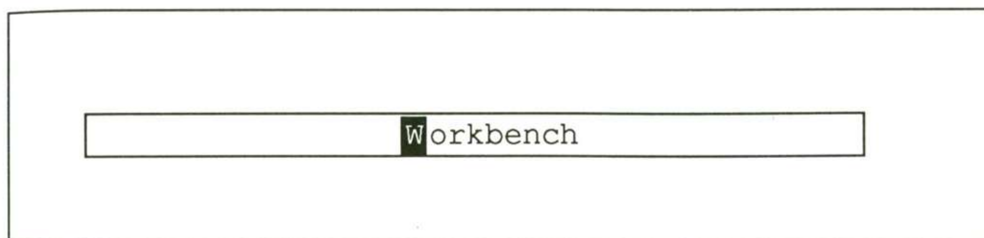


Bild 6.13: Ein Text-Gadget

Um Text in ein Text-Gadget eingeben oder ändern zu können, müssen Sie es in der Regel zuerst einmal mit der Maus anklicken. Sobald dies geschehen ist, erscheint auch die Schreibmarke, ein Buchstabe (es kann auch eine Leerstelle sein), der in anderen Farben erscheint als seine Umgebung, der also »hervorgehoben« wird. Das ist die Schreibmarke des Text-Gadgets. Wenn Sie jetzt Text eintippen, so erscheint dieser links vor der Schreibmarke. Jeder Buchstabe, den Sie neu eingeben, schiebt die Zeichen rechts neben der Schreibmarke eine Stelle weiter nach rechts.

Wollen Sie einen bereits in einem Text-Gadget stehenden Text ändern oder haben Sie einen Fehler gemacht und wollen diesen korrigieren, müssen Sie meistens die Schreibmarke erst zu der fehlerhaften Stelle bewegen. Hierzu dienen die Pfeiltasten, die die Schreibmarke in die auf der Taste aufgedruckte Richtung bewegen – solange es geht. Ist das Text-Gadget nur eine Zeile hoch, dann bewirken die beiden Pfeiltasten, die nach oben und nach unten weisen, natürlich nichts. Bei mehrzeiligen Text-Gadgets allerdings kann man mit ihnen auch die Zeile wechseln.

Wenn Sie rasch zum Anfang oder Ende des Textes in einem Text-Gadget gelangen wollen, können Sie dazu die Tastenkombinationen [Sh]+[<—] (Sprung zum Zeilenanfang) und [Sh]+[—>] (Sprung zum Zeilenende) verwenden. Ein anderes Hilfsmittel zum schnellen Verschieben der Schreibmarke ist auch die Maus. Wenn Sie einen Buchstaben in einem Text-Gadget nämlich anklicken, springt die Schreibmarke sofort zu dieser Stelle.

Sind Sie unter Verwendung der Maus oder der Pfeiltasten an der Stelle angekommen, an der Sie Änderungen vornehmen wollen, so können Sie dort neuen Text einfügen (indem Sie diesen einfach eintippen) oder auch löschen. Zum Löschen dienen die beiden Tasten [Backspace] und [Del] rechts oben auf der Tastatur. [Del] löscht den Buchstaben, über dem sich die Schreibmarke gerade befindet, wobei die rechts davon liegenden Buchstaben »nach links aufrücken«. Die [Backspace]-Taste löscht den Buchstaben links von der Schreibmarke, wobei die Buchstaben ab der Schreibmarke nach links aufrücken. Die [Backspace]-Taste dient vor allem zur sofortigen Korrektur von Eingabe-Fehlern, während man mit der [Del]-Taste gut alten Text in Text-Gadgets entfernen kann, bevor man neuen eintippt.

Besonders zur Korrektur größerer Fehler oder bereits in einem Text-Gadget stehenden Textes sind einige weitere Tastenkombinationen sehr nützlich. [*A*]+[X] löscht den gesamten Text, der in einem Text-Gadget steht, mit einem Tastendruck und [*A*]+[Q] stellt den Inhalt des Text-Gadgets wieder her, den es hatte, bevor Sie irgendwelche Änderungen daran durchgeführt

haben. (Zur Erinnerung: [*A*]+[X] bedeutet, daß Sie zunächst die Amiga-Taste drücken und sie festhalten müssen und dann die Taste [X] drücken.) Wenn Sie einen komplett neuen Text eingeben wollen, ist ein schnelles Löschen mit [*A*]+[X] zum Beispiel sehr sinnvoll. Und wenn Sie viele Änderungen durchgeführt haben und feststellen, daß Sie das gar nicht wollten, ist [*A*]+[Q] »Rettung in höchster Not«.

Ist der Text im Text-Gadget nach all diesen Änderungen zu Ihrer Zufriedenheit ausgefallen, sollten Sie die Eingabe immer noch durch Betätigung der [Return]-Taste (die große L-förmige Taste links von den Pfeiltasten) oder der [Enter]-Taste (rechts unten im Zahlenblock) beenden.

Wenn Sie häufig mit Texteingaben zu tun haben, werden Sie auch die Möglichkeit der automatischen Tastenwiederholung zu schätzen wissen. Halten Sie eine beliebige Taste längere Zeit fest, so wirkt dies nach einer kurzen Pause, als würden Sie diese Taste schnell hintereinander immer wieder betätigen. Wollen Sie zum Beispiel größere Textstellen löschen oder die Schreibmarke große Strecken bewegen, ist diese automatische Wiederholung sehr praktisch. Wie rasch die simulierten Tastendrücke erfolgen und wie lange die Pause dauert, bis die automatische Wiederholung beginnt, können Sie im Hauptfenster von *Preferences* einstellen.

6.4.4 Kombinierte Gadgets

Viele Gadgets, die wie ein Bedienungselement aussehen, sind in Wirklichkeit aus mehreren Intuition-Gadgets zusammengesetzt. Die meisten Schieberegler bestehen zum Beispiel aus mehreren Komponenten, von denen einige in die Kategorie Sensoren einzuordnen sind. Bei vielen Schiebereglern kann man den eigentlichen Regler nicht nur verschieben, indem man ihn mit der Maus ergreift, sondern auch, indem man in das Gebiet des Schiebereglers neben den Knopf des Reglers klickt. Dieser springt dann ein kleines Stück auf den Mauszeiger zu. Dieses Gebiet verhält sich also wie ein Schalter beziehungsweise Sensor.

Auch Rollbalken sind solche kombinierten Gadgets, und Ihnen werden in anderen Programmen bestimmt noch weitere Kombinationen dieser Art begegnen. Wenn Sie diese »neuen« Gadgets aber genau betrachten, werden Sie feststellen, daß Ihnen die Teile alle schon bekannt sind. Wenn man auf diese Weise »durchschaut« hat, woraus ein Gadget besteht, kann man es meist auch sofort problemlos bedienen.

6.4.5 Merkwürdig aussehende Gadgets

Manche Gadgets, die am Bildschirm erscheinen, wenn Sie eine Speichererweiterung Amiga 501 besitzen, sehen »merkwürdig« aus. Besser gesagt: sie sehen nach überhaupt nichts bestimmtem, sondern wie eine zufällige Wolke von Punkten aus. Solche Gadgets kommen zustande, wenn Sie mit einem Programm arbeiten, das geschrieben wurde, als es noch keine Amigas mit mehr als 512 Kbyte RAM-Speicher gab. Die ersten 512 Kbyte RAM-Speicher sind nämlich ganz besonderer Speicher (der sogenannte »Chip-RAM«). In diesem Bereich müssen alle Dinge wie Bilder, Screens, Fenster, Klangbeschreibungen und so weiter abgespeichert werden, auf die die Custom-Chips des Amiga Zugriff haben. Fehlerhaft programmierte

Programme legen aber manchmal solche Dinge in den Bereich oberhalb der ersten 512 Kbyte RAM, was auf dem Bildschirm die oben beschriebenen merkwürdigen Effekte ergibt.

Das ist eigentlich nur ein »Schönheitsfehler«, den Sie nicht unbedingt weiter beachten müssen. Wenn ein Programm allerdings solche Schönheitsfehler besitzt, ist es wahrscheinlich, daß es noch weitere ähnliche Fehler besitzt, die vielleicht schwerwiegendere Konsequenzen haben. Benutzen Sie ein Programm, das zufällige Punktwolken am Bildschirm zeigt, wo eigentlich (laut Handbuch) Symbole oder Bilder erscheinen sollten, mit äußerster Vorsicht!

6.5 Requester

Vielleicht ist Ihnen schon einmal aufgefallen, daß Gadgets üblicherweise gehäuft in einer ganz bestimmten Sorte von Fenstern auftauchen. Diese Fenster erscheinen mehr oder weniger unverhofft auf dem Bildschirm, sobald Sie einen bestimmten Befehl aus einem Menü ausgewählt haben, und verlangen eine Eingabe oder Entscheidung von Ihnen. Sobald sie diese Eingabe erhalten haben, verschwinden sie (schließen sich) von selbst wieder.

Bei solchen Fenstern handelt es sich meist um sogenannte »Requester«. Der Name kommt daher, daß ein Programm etwas von Ihnen verlangt (engl. *to request = verlangen/fordern*), wenn ein Requester auftaucht. Requester sind ebenfalls ein besonderes Intuition-Konzept, da sie sich teilweise von »normalen« Fenstern unterscheiden.

6.5.1 Erscheinen von Requestern

Requester tauchen immer dann auf, wenn ein Programm eine Entscheidung oder zusätzliche Informationen von Ihnen benötigt. Ein Beispiel für einen sehr simplen Requester ist das einsame Text-Gadget, das erscheint, wenn man auf der Workbench den Befehl *Rename* aus dem *Workbench*-Menü wählt. In diesem Fall wird ein neuer Name für ein Piktogramm benötigt. Solche (und natürlich auch komplexere) Requester tauchen meist in der Folge der Auswahl eines Menübefehls auf. Einige Programme verbinden einen bestimmten Requester auch mit einem Doppelklick der Menütaste. Wenn Sie wissen wollen, ob ein bestimmtes Programm auf einen Menütasten-Doppelklick reagiert, probieren Sie es einfach aus (die meisten Programme haben aber keinen solchen Requester).

6.5.2 Fehlermeldungen

Eine häufige Anwendung für Requester besteht auch in der Ausgabe von Fehlermeldungen. Ein neues Fenster am Bildschirm fällt auf und ist somit ideal dazu geeignet, die Aufmerksamkeit des Anwenders auf eine Fehlermeldung zu richten. Solche Fehlermeldungen müssen meist durch einen Mausklick in ein Tasten-Gadget bestätigt werden, bevor sie wieder verschwinden. Das Progamm weiß dadurch, daß der Anwender die Meldung zur Kenntnis genommen hat. Manchmal haben Sie verschiedene Optionen, nach einem Fehler fortzufahren. Ein Requester ist eine gute Möglichkeit, dem Anwender diese Optionen anzubieten, damit nicht das Programm, sondern er darüber entscheiden kann.

Viele System-Requester sind ein gutes Beispiel für die gerade beschriebene Nutzung von Requestern. Sie heißen deshalb »System-Requester«, weil sie nicht von einem bestimmten Anwendungsprogramm geöffnet werden, sondern von der Systemsoftware des Amiga 500.

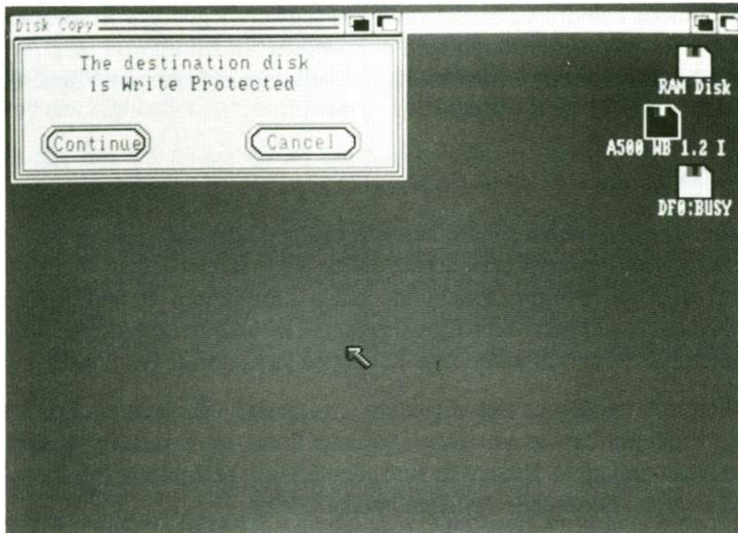


***Bild 6.14:** Ein System-Requester*

Versucht man zum Beispiel, einen Schnappschuß (mit dem Befehl *Snapshot*) von einem Objekt zu machen, während die Diskette, auf der sich das Objekt befindet, mit einem Schreibschutz versehen ist, so erscheint ein System-Requester mit entsprechender Meldung, der zugleich zwei Optionen (mit zwei Tasten-Gadgets) anbietet. Klickt man auf die Taste *Cancel*, so wird der Snapshot-Befehl abgebrochen. Wollen Sie aber, daß der *Snapshot*-Befehl auf alle Fälle ausgeführt wird und hatten nur vergessen, den Schreibschutz zu entfernen, so können Sie die Diskette aus dem Laufwerk nehmen, den Schreibschutz beseitigen, die Diskette wieder einlegen und dann den Knopf *Retry* betätigen. (Wenn Sie nicht schnell genug beim Betätigen des Knopfes sind, versucht es der Amiga 500 von selbst noch einmal.)

6.5.3 Requester und Fenster

Die meisten Requester sehen auf den ersten Blick aus wie ganz »gewöhnliche« Fenster, die allerdings manchmal auffällige Farben tragen. Sie können Schließ-, Front-, Back- und (seltener) Größen-Gadgets besitzen, und fast alle Requester können auch auf dem Bildschirm hin und her bewegt werden, indem man sie an ihrer Titelleiste ergreift.

Genau wie bei anderen Fenstern speichert Intuition zu den Requestern auch ab, zu welchem Programm sie gehören. Klickt man deshalb in ein Requester-Fenster, das inaktiv war, so wird

das zugehörige Programm ebenfalls aktiv und übernimmt zum Beispiel die Titelleiste des Bildschirms. Genauso kann man einen auftauchenden Requester auch manchmal ignorieren und zunächst mit einem anderen Programm weiterarbeiten, während man sich eine Antwort auf den Requester überlegt. Hierzu braucht man nur in ein Fenster (oder einen Screen) eines anderen Programms zu klicken.

6.5.4 Besonderheiten von Requestern

Requester haben jedoch einige wesentliche Besonderheiten, die sie von normalen Fenstern deutlich abheben. Dies sind zunächst einmal reine Äußerlichkeiten. Außer den üblichen Gadgets, die die meisten Fenster enthalten, besitzen die meisten Requester nämlich mindestens noch ein oder zwei beschriftete Knöpfe. Eine Betätigung (Anklicken) dieser Knöpfe schließt den Requester zumeist und teilt dem Programm gleichzeitig Ihre Entscheidung mit – sofern mehrere Tasten zur Auswahl standen.

Typisch ist oft eine Auswahl zwischen den Tasten OK (weitermachen; alles klar) und CANCEL (Abbrechen; alles vergessen). OK dient typischerweise zur Bestätigung aller Eingaben, die man in dem Requester gemacht hat, und CANCEL als Notbremse, wenn man »es sich anders überlegt hat« und der Amiga 500 den vorangegangenen Befehl ignorieren soll.

Neben diesen Äußerlichkeiten gibt es noch einen sehr wichtigen Aspekt des Verhaltens von Programmen, solange ein Requester dieses Programms geöffnet ist. Das Programm befindet sich nämlich in einem Wartezustand, solange der Requester nicht beendet wurde. Klickt man auf ein Fenster des Programms, dann ändert der Mauszeiger seine Form zur ZZ-Sprechblase und ignoriert alle Eingaben innerhalb des Fensters. Drückt man auf die Menütaste, wird die Titelleiste des Bildschirms nur weiß und es erscheinen keine Menüs. Der Requester muß erst beendet (geschlossen) werden, bevor mit diesem Programm weitergearbeitet werden kann. Dieses Verhalten (der Wartezustand des Programms) ist auch verständlich. Der Requester ist ja ein Zeichen dafür, daß das Programm Informationen oder Entscheidungen benötigt, bevor es fortfahren kann. Diese Entscheidung kann nicht durch einen Trick umgangen werden.

Mit anderen Programmen kann man allerdings weiterarbeiten, solange kein Requester für sie offen ist. Klickt man auf das Fenster eines anderen Programms, normalisiert sich der Mauszeiger wieder (falls er überhaupt in Schlafstellung war) und drückt man die Menütaste, so erscheinen auch die Menüs des anderen Programms. Ein Requester »sperrt« nur ein Programm vor der Benutzung.

6.5.5 Alerts

Eine besondere Art von Requestern sind die Alerts. Sie werden im Gegensatz zu den Requestern nur für wirklich katastrophale Fehlermeldungen verwendet, bei denen keine Rettung mehr möglich ist. Bei einem Alert wird der ganze Bildschirm oder ein Teil des Bildschirms pechschwarz, und mit roter Schrift erscheint die Fehlermeldung am oberen Bildschirmrand. Das sieht recht beängstigend aus und ist es meist auch, weil es auf einen Fehler hinweist, bei dem eigentlich kein Ausweg mehr möglich ist. Es ist im allgemeinen ein Fehler des Programmierers der entsprechenden Software gewesen, der eine mögliche Programm-

situation nicht berücksichtigt hat. (Regel Nummer 1 des Computer-Anwenders: Kein Programm – auch nicht das kleinste – ist fehlerfrei!)

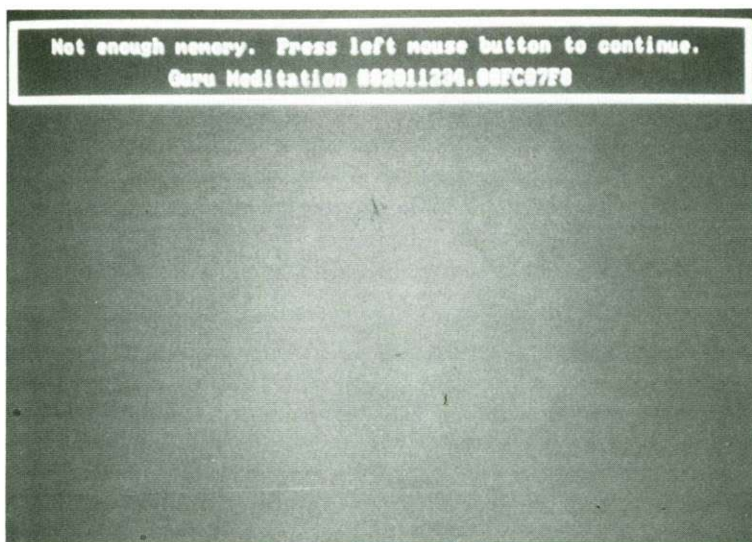


Bild 6.15: Ein Alert

Alerts fordern Sie zumeist auf, eine der beiden Maustasten zu drücken, bevor Sie fortfahren. Bevor Sie dies tun, vergewissern Sie sich bitte, daß die Diskettenlaufwerke nicht aktiv sind, und nehmen Sie alle Disketten aus den Laufwerken. Drücken Sie dann auf die angegebene Maustaste. Da Alerts meist englisch sind, heißt die linke Maustaste in einem Requester *left mousebutton* und die rechte *right mousebutton*.

Begegnet Ihnen in einem bestimmten Programm immer wieder ein bestimmter Alert, so ist es ganz ratsam, sich die beiden Nummern zu notieren, die Ihnen (im Alert) hinter dem Ausdruck Guru Meditation Number genannt wird. Bitten Sie dann Ihren Händler, Ihnen eine funktionierende Version dieses Programms zu beschaffen und geben Sie ihm den Zettel mit dieser Zahl. Der Programmierer, der das fehlerhafte Programm geschrieben hat, kann daraus unter Umständen schließen, welchen Fehler er gemacht hat. Sie können durch das Notieren der entsprechenden Zahl eventuell dazu beitragen, daß schneller eine neue, fehlerfreie Version des Programms entsteht.

6.6 Screens

Mausbedienung, Pull-down-Menüs und Fenster gibt es inzwischen auch auf anderen Computern. In dieser Beziehung steht der Amiga 500 beziehungsweise Intuition also nicht mehr ganz allein. Eine Fähigkeit des Amiga ist jedoch einzigartig und wird wohl auch nicht so schnell kopiert werden. Das sind die »virtuellen Bildschirme« oder »Screens« von Intuition.

Die Grafik-Hardware des Amiga erlaubt es nämlich, jedem Programm auf Wunsch einen eigenen virtuellen Bildschirm zur Verfügung zu stellen, über dessen Eigenschaften das Programm völlig frei verfügen kann, ohne andere Programme dadurch zu beeinträchtigen. So können zum Beispiel die Anzahl und die Farbtöne der Farben, die auf dem Bildschirm für Abbildungen zur Verfügung stehen, frei gewählt werden. Auch die horizontale und vertikale Auflösung des Bildschirms kann bei unterschiedlichen Screens unterschiedlich sein, und so weiter. Zwischen diesen verschiedenen virtuellen Bildschirmen können Sie als Anwender entweder mit einem schnellen Tastendruck hin und her wechseln oder Sie können den wirklichen Bildschirm (man sagt auch »physikalischen Bildschirm«) in verschiedene Bereiche aufteilen, in denen Sie Teile der verschiedenen Screens gleichzeitig betrachten können. Wie groß der Anteil welches virtuellen Bildschirms am wirklichen Bildschirm ist, hängt nur von Ihren Wünschen und dem freien Speicherplatz in Ihrem Amiga 500 ab.

6.6.1 Bedienung von Screens

Virtuelle Bildschirme verhalten sich im wesentlichen wie Fenster. Mit dem großen Unterschied allerdings, daß man ihre Größe nicht ändern und sie nicht horizontal verschieben kann. Mehrere Screens liegen zum Beispiel auch in der Art eines Stapels übereinander, wobei oben liegende die darunter liegenden (teilweise) verdecken. Erst wenn der obenliegende Screen nicht mehr den ganzen Bildschirm bedeckt, ist es überhaupt möglich, andere Screens zu sehen.

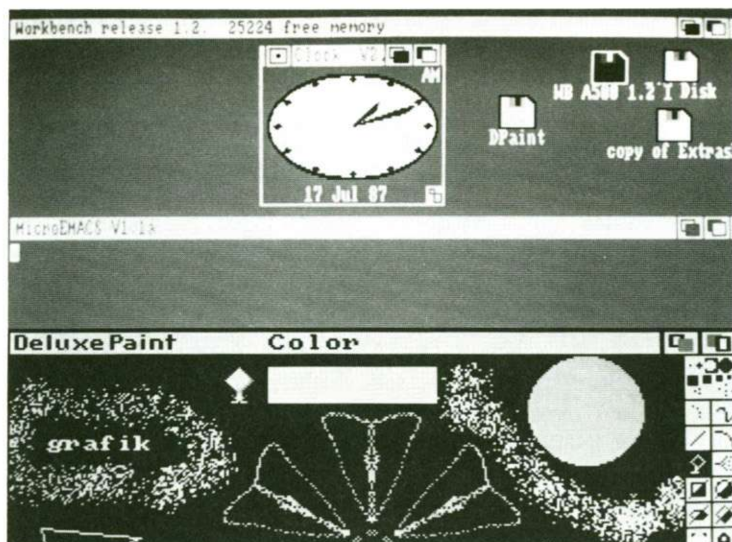


Bild 6.16: Mehrere Screens am Bildschirm

Genau wie bei Fenstern, kann man auch bei Screens die Reihenfolge im Stapel ändern. Hierzu dienen das Back-Gadget und das Front-Gadget am rechten Rand der Titelleiste eines Screens. Diese Gadgets sehen genauso aus wie die entsprechenden Fenster-Gadgets – und funktionieren auch genauso. Betätigt man das Front-Gadget eines Screens, wird dieser der oberste. Er kommt also ganz nach vorn und verdeckt dabei eventuelle andere Screens. Betätigt man das Back-Gadget, wird der entsprechende Screen der unterste (und ist dadurch vielleicht nicht mehr zu sehen, weil er von anderen Screens verdeckt wird).

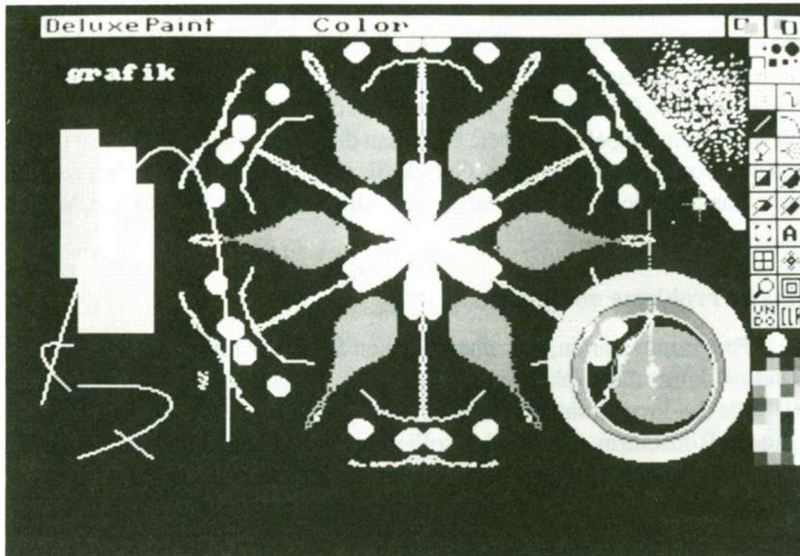


Bild 6.17a: Vor der Betätigung des Back-Gadgets an einem Screen

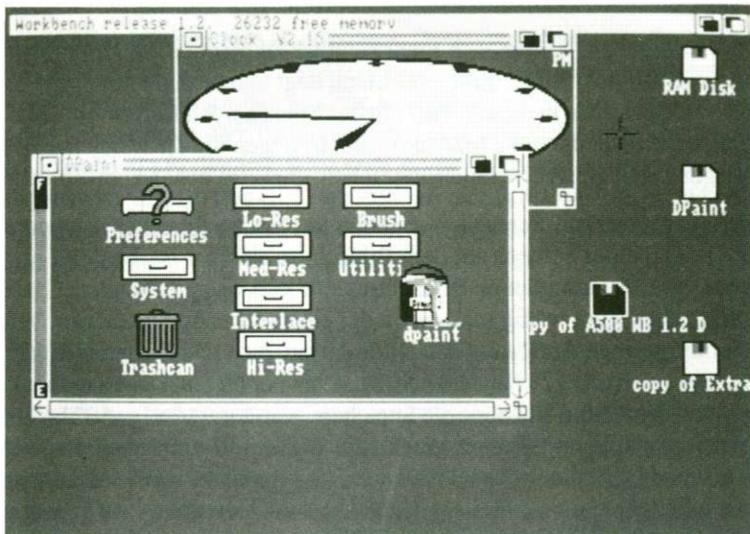


Bild 6.17b: Nach der Betätigung des Back-Gadgets an einem Screen

Da die Workbench immer ein besonders wichtiger Screen ist und einige Programme das Umlegen von Screens möglicherweise nicht korrekt durchführen, gibt es für die Workbench

noch besondere Tastenkombinationen, die Sie entweder ganz nach vorne oder ganz nach hinten legen – egal ob die dafür eigentlich zuständigen Gadgets gerade sichtbar sind oder nicht. (Es gibt Programme, die eigene Screens öffnen und diese nicht mit Back- oder Front-Gadgets versehen.) Hält man die Commodore-Taste fest und drückt dann die Taste [N], so kommt der Workbench-Screen im Stapel nach oben. Hält man die Commodore-Taste fest und drückt auf [M], so wird er ganz nach hinten gelegt. Gerade die erste Tastenkombination ist sehr wichtig, da sie Ihnen eine Möglichkeit verschafft, »auf alle Fälle« zurück auf die Workbench zu gelangen.

6.6.2 Mögliche Probleme mit Screens

Wenn Sie ein Programm starten, das einen eigenen Screen benötigt, so sorgt das Programm meist von selbst dafür, daß dieser Screen nach dem Start der oberste und damit komplett sichtbar ist. Falls das Programm das nicht tut, nach dem Programmstart also scheinbar keine Veränderung am Bildschirm zu sehen ist, sollten Sie versuchen, den Workbench-Bildschirm (mit dem entsprechenden Gadget) nach hinten zu legen. Falls keine anderen virtuellen Bildschirme existieren, wird der Screen des neuen Programms dadurch sichtbar.

Wenn die Front- und Back-Gadgets eines Screens scheinbar nicht funktionieren, hat der Programmierer des entsprechenden Programms wahrscheinlich einen Fehler gemacht. Bei einigen Demo-Programmen können zum Beispiel solche Effekte auftreten. Versuchen Sie dann mit den entsprechenden Tasten-Kombinationen ([*C*=]+[N] oder [*C*=]+[M]), die Workbench nach vorne zu holen beziehungsweise nach hinten zu legen.

Scheint auch [*C*=]+[N] die Workbench nicht sichtbar zu machen, so kann es sein, daß der Workbench-Screen auf dem Bildschirm ganz unten liegt und deshalb nur ein kleiner Streifen davon sichtbar ist. Gehen Sie in diesem Fall mit der Maus am Bildschirm so tief herunter wie möglich, drücken Sie dann die linke Maustaste und bewegen Sie die Maus wieder nach oben. Das sollte, von ganz wenigen Ausnahmen abgesehen, die Workbench zum Vorschein bringen.

Das größte Problem, das mit Screens verbunden ist, ist ihr Speicherplatzbedarf. Obwohl vielleicht nur ein Teil eines Screens auf dem Bildschirm sichtbar ist, benötigt er doch immer soviel Speicher, wie ein kompletter Bildschirm mit den entsprechenden Grafikattributen benötigen würde. Dies kann wenig sein, wenn der entsprechende Screen nur wenige Farben zeigt und/oder eine geringe Auflösung hat. Bildschirme, die viele Farben gleichzeitig zeigen, können bis zu etwas mehr als 128 Kbyte RAM-Speicher benötigen. Das ist mehr als ein Viertel des Speichers, den der Amiga 500 ohne die Speichererweiterung Amiga 501 besitzt. Seien Sie deshalb immer vorsichtig mit Programmen, die eigene Bildschirme mit vielen Farben benötigen. Versuchen Sie nie, mit mehr als ein oder maximal zwei solchen Programmen gleichzeitig zu arbeiten.

7 Einführung in das CLI

Bis jetzt wurde in diesem Buch nur eine der Möglichkeiten, den Amiga zu bedienen, behandelt: die Workbench. Die Workbench erlaubt es, unter Nutzung der Benutzerschnittstelle *Intuition*, alle Aufgaben, die bei der Bedienung eines Computers anfallen, auf sehr komfortable Weise zu erledigen. Der Inhalt der Computerspeicher, vor allem der Disketten und Festplatten, wird in grafischer Form verdeutlicht. Alle manipulierbaren Objekte werden in Form kleiner Bilder, der Piktogramme, dargestellt und die wichtigsten Operationen sind oft durch ein, zwei Mausklicks und Mausbewegungen zu erledigen. Diese freundliche, grafische Darstellung der komplexen Vorgänge unterscheidet den Amiga von vielen anderen Computern, bei denen man die Befehle in Form schwer zu behaltender Kommandos über die Tastatur eingibt und die Ergebnisse dann oft als lange Textlisten zu sehen bekommt. Der Amiga besitzt aber auch noch eine andere Methode der Bedienung (eine andere »Benutzerschnittstelle«, wie der Fachmann sagt): das sogenannte »CLI«. Das CLI repräsentiert eine Art, den Amiga zu bedienen, die der »üblichen« Bedienungsweise eines Computers, wie man es von Modellen anderer Hersteller kennt, recht nahekommt.

7.1 Das CLI

»CLI« steht für *Command Line Interpreter* oder *Command Line Interface*. Das konnte man in etwa mit *Kommandozeilen-Ausführer* oder *Kommandozeilen-Schnittstelle* übersetzen; die Amiga-Dokumentation und die Belegschaft von Commodore Deutschland ist sich über den genauen Ursprung der Abkürzung etwas uneins. Ich werde im folgenden aber sowieso immer nur die Abkürzung CLI verwenden – sie ist einfach wesentlich kürzer als die beiden kompletten Namen. Die Kommunikation über diese Schnittstelle spielt sich über die Tastatur und über ein Fenster ab, das sich verhält, als wäre es ein Computer-Terminal, das nur Text darstellen kann. Die Grafikmöglichkeiten des Amiga liegen hier völlig brach und auch die Maus kann nicht benutzt werden!

»Wozu dies alles? Wozu die vielen Möglichkeiten des Amiga ungenutzt lassen?«, werden Sie jetzt vielleicht fragen. Aber auch diese zweite Art, den Amiga zu bedienen, hat ihre Berechtigung, wie wir in den folgenden Kapiteln noch sehen werden. Die Workbench hat einige Einschränkungen und Umständlichkeiten, die Ihnen jetzt wahrscheinlich noch nicht auffallen werden. Vielleicht werden Sie diese Einschränkungen sogar nie bemerken oder nicht als solche empfinden. Bei einem täglichen Gebrauch und bei speziellen Anwendungen, wie zum Beispiel der Programmierung, können sie aber bald lästig werden. Deshalb gibt es das CLI. Es mag schwieriger zu erlernen sein, ist aber sehr leistungsfähig.

7.1.1 Was ist das CLI?

Das CLI ist ein Programm (fast) wie jedes andere. Computer-Wisssenschaftler würden es eine *Shell* (zu deutsch *Schale*) nennen. Startet man es, so legt es sich wie eine Schale (daher der Name) um den Betriebssystemkern des Amiga. Dieser Kern ist ein recht komplexes Software-Gebilde, mit dem zu kommunizieren den wenigsten Anwendern leichtfallen dürfte. Für die Mehrzahl der Anwender, die nicht in der binären Computersprache denken, ist eine Shell da. Die Befehle des Benutzers werden von einer solchen Shell aus ihrer menschenverständlichen und einigermaßen freundlichen Form in die Sprache überführt, die der Computer versteht. Genauso werden die Ergebnisse der Befehle, die der Benutzer erteilt und die vom Betriebssystemkern ausgeführt wurden, zunächst von der Shell empfangen und in eine Sprache übersetzt, die der menschliche Anwender verstehen kann. Bei der Workbench ist dies eine bildliche Sprache, beim CLI eine etwas schwierigere, textuelle Sprache. Aber auch sie ist erlernbar – und zwar leichter als die meisten menschlichen Sprachen.

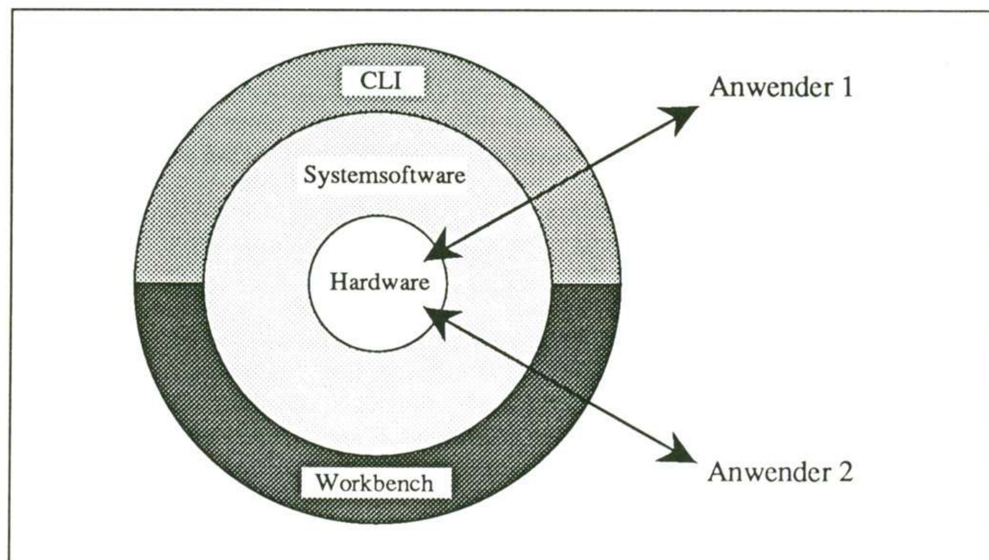


***Bild 7.1:** Zwei verschiedene Shells, die den »Kern« des Computers umgeben*

Prinzipiell haben die meisten Computer eine »bevorzugte« Shell, mit der ein Anwender »normalerweise« arbeitet; beim Amiga ist es die Workbench. Bei vielen Computern ist es aber auch problemlos möglich, sie durch ein anderes Programm zu ersetzen oder zu ergänzen. Das bekannte Computer-Betriebssystem UNIX™, von dem Sie vielleicht schon gehört haben, aber auch der Amiga nutzen diese Möglichkeit, die Shell nach Wunsch auszuwechseln oder gar mit mehreren Shells gleichzeitig zu arbeiten. Sie als Anwender können immer die Shell wählen, die Ihnen am besten gefällt oder die für eine bestimmte Aufgabe am besten geeignet erscheint.

Das CLI ist mehr für den fortgeschrittenen Benutzer gedacht, der schon eine Weile mit dem Amiga gearbeitet hat. Wer gerade erst seinen Amiga gekauft hat und die verschiedenen Optionen der Workbench schon verwirrend genug findet, der sollte das Studium dieses Buchteils vielleicht auf einen späteren Zeitpunkt verschieben.

7.1.2 Was ist Amiga-DOS?

Im Zusammenhang mit dem CLI hört man oft auch noch den Begriff »Amiga-DOS«, der manchmal auch mit dem CLI verwechselt wird. Amiga-DOS steht für *Amiga-Disk-Operating-System* und ist ein (wichtiger) Teil des Betriebssystemkerns, der weiter oben schon angesprochen wurde. Amiga-DOS kümmert sich zum Beispiel um die Verwaltung der Diskettenlaufwerke und anderer Geräte und macht es den eigentlichen Anwendungsprogrammen (und dem Computer-Benutzer) leichter, mit diesen Geräten zu arbeiten.

Das CLI mit Amiga-DOS zu verwechseln ist aber andererseits auch einfach, da das CLI einen direkten Zugriff auf nahezu alle Möglichkeiten von Amiga-DOS bietet. Jedes »Objekt«, wie zum Beispiel eine Datei, das Amiga-DOS erkennt und manipulieren kann, kann man auch mit dem CLI manipulieren. Und zu nahezu jeder Routine (jedem Programmteil) von Amiga-DOS gibt es einen CLI-Befehl, der dieser Routine fast 1:1 entspricht.

7.2 Wie Sie das CLI starten

Kommen wir nun zur praktischen Arbeit mit dem CLI. In diesem Kapitel werden Sie Ihre ersten zaghaften Schritte mit dieser zunächst unfreundlich erscheinenden Art, Ihren Amiga 500 zu bedienen, machen. Wenn Sie bereits andere Computer kennen – speziell UNIX- und MS-DOS-Systeme – werden Ihnen viele Aspekte beim Umgang mit dem CLI bekannt vorkommen. Sie werden dann vielleicht versucht sein, dieses Kapitel sehr flüchtig oder gar nicht zu lesen. Tun Sie das bitte nicht. Sie werden in den folgenden Abschnitten nämlich einige wichtige Grundlagen des CLI kennenlernen, die nicht, oder nicht unbedingt in dieser Form auf anderen Computern gültig sind.

7.2.1 Vorarbeiten für die Arbeit mit dem CLI

Bevor Sie mit den ersten Versuchen beginnen, sollten Sie aber zunächst erst eine Kopie der originalen Workbench-Diskette anfertigen. Wie Sie eine solche Kopie anfertigen (auf der Workbench), wird im zweiten Kapitel dieses Buches ausführlich beschrieben. Wenn die Kopie fertig ist, beschriften Sie sie sinnigerweise mit dem Namen »CLI-Diskette«. Geben Sie ihr auch

auf der Workbench mit dem *Rename*-Befehl den Namen »CLI-Diskette«). Diese Diskette werden Sie für den Rest dieses Buchteils für alle Experimente verwenden. Die erste Kopie der Workbench-Diskette legen Sie bitte so lange erst einmal beiseite. Sind Sie mit dem Kopieren fertig, so nehmen Sie bitte beide Disketten aus den Laufwerken und starten den Amiga 500 neu (mit der Tastenkombination [Ctrl]+[*C*=]+[*A*]). Schieben Sie dann die neu kopierte Diskette in das interne Laufwerk des Amiga 500. Für den Rest dieses und den größten Teil der folgenden Kapitel nehme ich an – sofern nicht ausdrücklich etwas anderes erwähnt wird – daß Sie nur mit dieser Diskette arbeiten.

7.2.2 Aktivieren des CLI

Es ist für Sie vielleicht gar nicht so einfach, das CLI zu starten. Nach dem Einschalten des Amiga und dem Einlegen der Workbench-Diskette erscheint ja sofort die Workbench und ein » CLI« ist weit und breit nicht zu sehen. Öffnen Sie nun die Workbench-Diskette mit einem Doppelklick auf ihr Piktogramm. Eventuell ist nun ein Piktogramm namens *CLI* im Diskettenfenster zu sehen. »Eventuell« deshalb, weil es verschiedene Versionen der Workbench-Diskette gibt, die mit dem Amiga 500 ausgeliefert wurden. Bei einigen taucht das CLI im Diskettenfenster der Workbench-Diskette auf, bei anderen nicht. Falls es in Ihrem Fall nicht sofort zu sehen ist, liegt es vielleicht in der System-Schublade. Öffnen Sie deshalb die System-Schublade und suchen Sie auch darin nach einem Piktogramm namens *CLI*. Wenn Sie das CLI aber auch hier noch finden können, müssen Sie »etwas weiter ausholen«. Schließen Sie dazu das Fenster der System-Schublade wieder und starten Sie das Werkzeug *Preferences*.

Betrachten Sie dann genau das Hauptfenster von *Preferences*. In der unteren Hälfte am rechten Rand dieses Bildschirms werden Sie neben der Beschriftung *CLI* zwei Knöpfe entdecken. Das ist der CLI-Schalter. Dieser Schalter steht normalerweise auf *Off* (engl. für *aus*). Klicken Sie nun auf *On* und verlassen Sie *Preferences* mit einem Klick auf den Knopf *Save*. Warten Sie einige Augenblicke, bis Sie zurück in der Workbench sind, also wieder Disketten- und Schubladen-Piktogramme erscheinen und öffnen Sie die System-Schublade erneut. Im Fenster, das sich nun öffnet, sollte ein Piktogramm namens »CLI« erscheinen. Dies ist das Programm CLI, die zweite Benutzerschnittstelle des Amiga, mit der wir uns im Rest dieses Buchteils befassen werden. Sehen Sie jedoch kein Icon namens *CLI*, starten Sie bitte noch einmal das Programm *Preferences* und vergewissern Sie sich, daß der Schalter *CLI* auf *On* steht. Klicken Sie auf *Save*, warten einen Moment und führen einen Neustart (mit [Ctrl]+[*C*=]+[*A*]) durch. Öffnen Sie erneut die System-Schublade, wenn die Workbench wieder erschienen ist.

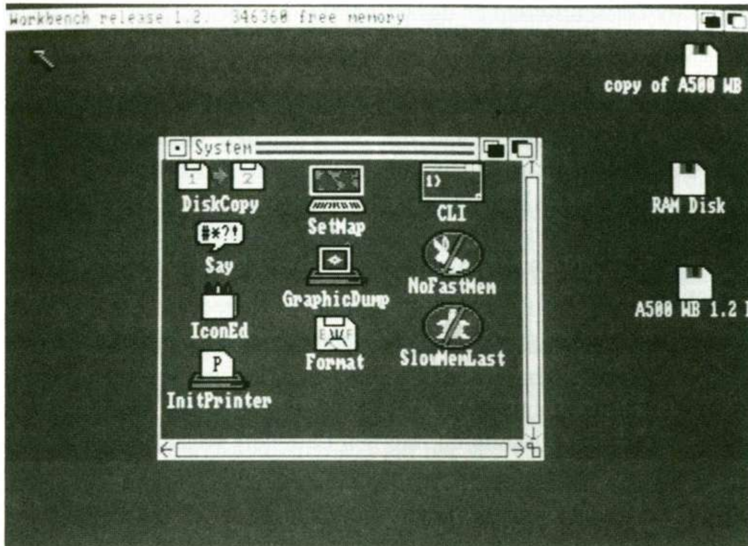


Bild 7.2: Das CLI-Piktogramm im System-Fenster

Ergreifen Sie nun das CLI-Piktogramm mit der Maus und legen Sie es in das Disketten-Fenster der Workbench-Diskette. Schieben Sie es an eine Stelle, an der es einigermaßen auffällig erscheint und wählen Sie den Befehl *Snapshot* aus dem Menü *Special*. Das CLI-Icon erscheint auch später immer wieder an dieser Stelle.

7.2.3 Erste Schritte im CLI

Starten Sie das CLI (indem Sie es mit einem Doppelklick auf das Piktogramm öffnen) und ein neues Fenster erscheint auf dem Bildschirm. Oben links in diesem Fenster erscheint der Text

```
1>
```

Dies ist das sogenannte »Prompt« des CLI, es erscheint immer dann, wenn das CLI bereit ist, einen neuen Befehl von Ihnen entgegenzunehmen und auszuführen.

Spätestens jetzt wird es Zeit für Sie, die Tastatur Ihres Amiga 500 einmal näher zu betrachten. Denn innerhalb des CLI-Fensters zählt nur die Tastatur, die Maus hat keine Funktion mehr. Sie können mit ihr zwar noch wie gewohnt die Position und Größe des Fensters ändern, ansonsten befinden Sie sich nun aber in einer Umgebung, die in gewisser Hinsicht vorsintflutlich ist und in der die Maus nicht zählt. Benutzer fast aller anderen Computer-Modelle müssen sich allerdings fast immer so ähnlich mit ihren Computern herumschlagen, deswegen sollten Sie sich nicht entmutigen lassen. Manches ist zwar lästig, aber dafür sind mit dieser Art der Computerbedienung andere Dinge auch wieder einfacher.

Tippen Sie nun Ihren ersten Befehl an das CLI ein:

```
1>dir
```

Während Sie auf der Tastatur die Buchstaben »d«, »i« und »r« tippen, erscheinen diese nacheinander auf dem Bildschirm und die Schreibmarke wandert weiter nach rechts. (Erscheint das »d« nicht sofort im Fenster, nachdem Sie es gedrückt haben, klicken Sie bitte kurz mit der Maus ins Innere des Fensters und versuchen es noch einmal. Erscheint dann immer noch nichts, so dürfte Ihre Tastatur defekt sein.) Das ganze ist dem Eintippen eines neuen Namens für ein Piktogramm in der Workbench sehr ähnlich, wie Sie es nach dem Befehl *Rename* tun können.

Tippen Sie nun [Return]. Sie schließen damit die Befehlseingabe ab. (Jeder Befehl an das CLI muß mit [Return] abgeschlossen werden. Das CLI liest den von Ihnen eingegebenen Befehl und versucht ihn auszuführen. In diesem Fall war es der Befehl *dir*, der eine Liste von Datei- und Schubladen-Namen der Diskette im internen Laufwerk ausgeben sollte. Haben Sie den Befehl richtig getippt, so »rollt« diese Liste nun vor Ihnen im Fenster vorbei – manchmal ein, manchmal zwei Namen in einer Zeile. Währenddessen läuft das Diskettenlaufwerk andauernd und die grüne Lampe flackert, da das CLI die Namen der Dateien natürlich von der Diskette lesen. Ist die Liste am unteren Rand des Fensters angekommen, bewegt sich der Fensterinhalt um eine Zeile nach oben (die obere Zeile verschwindet dabei). Die nächsten Namen tauchen dann in der untersten, jetzt leeren, Zeile auf. Diesen Vorgang nennt man »Rollen« oder auch »Scrollen« des Fensters. Sie haben ihn ja schon bei den Workbench-Fenstern kennengelernt. Anders als bei diesen Fenstern ist alles, was über den oberen Fensterrand hinaus verschwunden ist, aber unwiederbringlich verloren. Sie können nicht zurückrollen, wie in der Workbench.

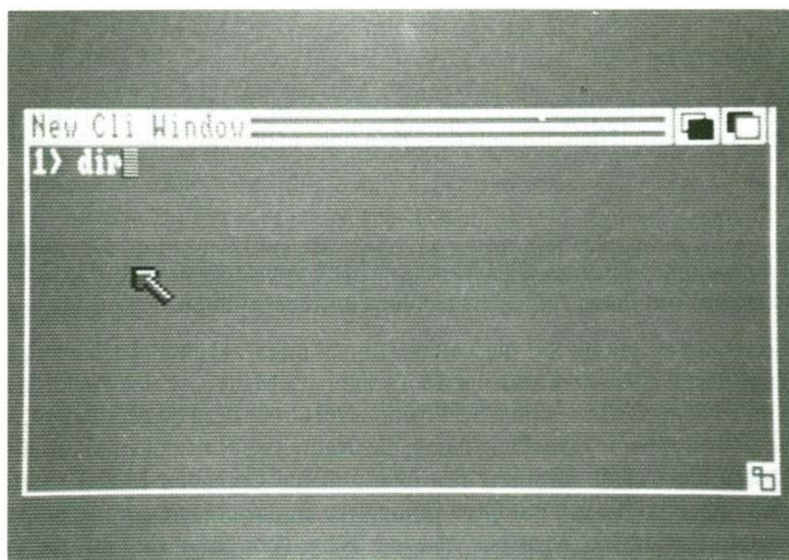


***Bild 7.3:** Eingabe eines Befehls im CLI*

Ist die Liste komplett, so erscheint wieder das Prompt und Sie können den nächsten Befehl eintippen:

```
1>
```

Nun erst ein paar Worte zur in diesem Buch verwendeten Schreibweise, bevor Sie mit der Arbeit fortfahren: Alles, was im CLI-Fenster erscheint – Ihre Eingaben und die Ausgaben des CLI – wird in diesem Buch mit `Schreibmaschinenschrift` gedruckt. Um zu unterscheiden, was Sie selbst eingeben müssen, werden diese Passagen fett hervorgehoben. Wenn Sie also `1>`**`dir`** lesen, so bedeutet das, daß im CLI-Fenster die Buchstaben »1>« erscheinen und Sie dahinter »dir« eintippen müssen. Und nicht vergessen: Wenn nicht ausdrücklich anders erwähnt, müssen Sie alle Eingaben im CLI-Fenster mit [Return] abschließen.

Nun aber zurück zu Ihrem ersten Befehl, den Sie gerade dem CLI erteilt haben. Manche der Namen, die in der Liste erscheinen, werden Ihnen bekannt vorkommen. Es sind dieselben, die auch einige Piktogramme im Diskettenfenster auf der Workbench tragen. Viele Namen aber werden neu sein, und das demonstriert schon einen wesentlichen Unterschied zwischen der Workbench und dem CLI. Im CLI können Sie alle Dateien und Schubladen (das CLI nennt Schubladen *Directories*; mehr dazu aber in einem späteren Abschnitt) sehen, die auf der Diskette liegen. In der Workbench sehen Sie hingegen nur die Dateien und Schubladen, von denen die Entwickler des Amiga meinen, daß Sie sie sehen sollten.

7.2.4 Wie Sie das CLI wieder verlassen

Wenn Sie zwischendurch auch einmal wieder in die Workbench oder gar das CLI endgültig verlassen wollen, bevor Sie das folgende Kapitel durcharbeiten, haben Sie mehrere Möglichkeiten: Wenn Sie nur vorübergehend etwas in der Workbench machen, und danach wieder zurück ins CLI wollen, ist es nicht nötig, dieses endgültig zu verlassen. Der Amiga besitzt ja ein Multitasking-Betriebssystem, was in diesem Fall bedeutet, daß beim Start des CLI die Workbench nicht beendet wurde. Die Workbench ist nach wie vor – neben oder hinter dem CLI – aktiv. Klicken Sie einfach mit dem linken Mausknopf in eines der Fenster oder auf eines der Piktogramme der Workbench. Sind die Fenster der Workbench nicht zu sehen, weil das CLI-Fenster sie alle verdeckt, so können Sie das CLI-Fenster natürlich auch wie jedes andere Fenster mit dem entsprechenden Gadget nach hinten legen.

Sobald Sie »zurück in der Workbench« sind, bemerken Sie, daß sich der Inhalt der Titelzeile des Bildschirms ändert. Wenn Sie den rechten Mausknopf niederdrücken, müssen nun auch wieder die Menüs der Workbench in der Menüzeile erscheinen. Sie können jetzt beliebige Aktionen in der Workbench durchführen und durch einen einfachen Mausklick auf das CLI-Fenster später dann wieder in das CLI zurückkehren.

Für Experimentierfreudige: Eine recht spektakuläre Möglichkeit, zu zeigen, daß Workbench und CLI wirklich unabhängig voneinander und gleichzeitig aktiv sein können ist, während der Ausführung eines CLI-Befehls in die Workbench zu gehen und dort zum Beispiel ein Piktogramm von einem Fenster in ein anderes zu legen. Der Befehl **dir** im CLI ist hierfür recht gut geeignet, da seine Ausführung meist eine ganze Weile dauert.

Eine andere Möglichkeit, das CLI zu verlassen, ist etwas permanenter. Sie beendet das Programm CLI wirklich und schließt auch das entsprechende Fenster. Falls das CLI-Fenster im Moment nicht aktiv ist, machen Sie es aktiv, indem Sie mit der Maus kurz in das Fensterinnere klicken. Tippen Sie dann hinter das Prompt:

```
1>endcli
```

Daraufhin erscheint eine kurze Meldung im Fenster, die aber im allgemeinen zu schnell verschwindet, als daß man sie lesen könnte, und das CLI-Fenster wird geschlossen. Die einzige Möglichkeit, jetzt wieder ins CLI zu gelangen, ist es, erneut das Piktogramm *CLI* zu öffnen. Der Befehl *endcli* ist also recht endgültig und sollte nur angewendet werden, wenn man sicher ist, das CLI in nächster Zeit nicht mehr zu benötigen. Er kann aber auch dann sehr sinnvoll sein, wenn der vom CLI benötigte Speicherplatz dringend für ein anderes Programm gebraucht wird.

7.3 Befehlseingabe

In diesem Abschnitt werden Sie etwas ausführlicher erfahren, was Sie bei der Eingabe eines Befehls an das CLI beachten müssen und welche Möglichkeiten Ihnen dabei zur Verfügung stehen.

7.3.1 Eingeben eines Befehls

Wie oben schon am Beispiel *dir* zu sehen war, bestehen CLI-Befehle immer aus einem (meist kurzen) Wort, das Sie eingeben müssen, damit das CLI eine bestimmte Operation ausführt. Welche Befehle Ihnen normalerweise zur Verfügung stehen, erfahren Sie noch in aller Ausführlichkeit im nächsten Kapitel. Eine Warnung aber vorweg: Es sind deutlich mehr als auf der Workbench – Sie müssen sie aber nicht alle kennen, um mit dem CLI arbeiten zu können!

Das CLI hat nämlich keinen »festen« Befehlssatz wie die Workbench. Wenn Sie einen Befehl eingeben, so sucht das CLI in Wirklichkeit nach einem Programm gleichen Namens, lädt dieses in den RAM-Speicher (sofern es gefunden wurde) und startet es. Indem Sie diese Befehls-Programme auf Ihrer CLI-Diskette löschen, umbenennen oder neue hinzufügen, können Sie also jederzeit den Befehlssatz des CLI ändern.

Jeder Befehl an das CLI besteht aus einem Befehlsnamen und einem oder mehreren (nicht immer notwendigen) »Parametern«. Der Befehlsname ist immer das erste Wort in der Zeile, die den Befehl enthält. Es ist zugleich auch der Name des Programms, nach dem das CLI sucht, um diesen Befehl ausführen zu lassen. Die Parameter sagen diesem Programm, was oder womit es etwas tun soll. Der Lösch-Befehl benötigt also zum Beispiel den Namen der zu löschenden Daten als Parameter.

7.3.2 Editieren einer Befehlszeile

Befehlszeilen im CLI können sehr lang werden, wenn ein Befehl viele Parameter hat. Sie müssen zudem auch sehr exakt eingetippt werden. Das CLI verzeiht nicht den kleinsten Fehler.

Es gibt leider viel zu wenige Möglichkeiten, die Zeile, die man gerade eingibt zu ändern – oder zu »editieren«, wie der Fachmann sagt.

Die wichtigste dieser Möglichkeiten stellt die [Backspace]-Taste dar. Sie löscht den zuletzt eingegebenen Buchstaben links von der Schreibmarke. Indem Sie mehrfach auf diese Taste drücken, können Sie auch mehrere Buchstaben nacheinander löschen – aber nur innerhalb der gerade bearbeiteten Zeile. Sie können mit [Backspace] nicht zurück in die vorangehende Zeile. Wenn Sie eine ganze Zeile löschen wollen, so können Sie das mit der Tastenkombination [Ctrl]+[X] (dazu müssen Sie die [Ctrl]-Taste festhalten und die [X]-Taste niederdrücken).

Die Tastenkombinationen und Mausklicks, mit denen Sie den Inhalt eines Text-Gadgets verändern können, haben (mit Ausnahme von [Backspace]) im CLI alle keine Wirkung!

7.3.3 Anhalten und Fortsetzen der Ausgabe

CLI-Befehle haben die unangenehme Eigenschaft, auch lange Texte in einem endlosen Strom auf den Bildschirm zu schreiben. Wird dabei das untere Ende des CLI-Fensters erreicht, rollt der alte Inhalt nach oben, um unten eine neue Zeile freizumachen. Das ist auch noch ganz sinnvoll. »Unangenehm an diesem Rollvorgang ist jedoch, daß Texte, die einmal am oberen Fensterand verschwunden sind, auch verschwunden bleiben und nicht wieder zurückgeholt werden können.

Sie können deshalb die Ausgaben eines Programms, die es in das CLI-Fenster schickt, jederzeit anhalten, um sich den Fensterinhalt in Ruhe zu betrachten, bevor er verschwindet. Drücken Sie hierzu einfach auf eine beliebige Taste (zum Beispiel die Leertaste). Drücken Sie danach auf [Backspace] oder [Return], wenn das Programm fortfahren soll. Probieren Sie es aus: Geben Sie dazu den folgenden Befehl ein:

```
1>dir
```

Warten Sie, bis einige Zeilen im Fenster erschienen sind und drücken Sie auf die Leertaste. Die Ausgabe hält dann sofort an – auf dem Bildschirm erscheinen keine neuen Zeichen. Es kann allerdings sein, daß das Diskettenlaufwerk noch einen Moment arbeitet. Sie können sich nun in Ruhe die bislang erschienene Liste betrachten. Drücken Sie nun die [Backspace]-Taste und (eventuell nach einer kleine Pause) erscheint neuer Text am Fenster, bis das Ende der Liste erreicht ist.

7.3.4 Abbrechen eines Befehls

Gelegentlich kann es nötig werden, in ein laufendes Programm einzugreifen. Eine dieser Möglichkeiten zur vorübergehenden Unterbrechung (mit anschließender Fortsetzung) der Ausgabe, haben Sie ja schon kennengelernt. Sie können aber Programme auch vom CLI aus abbrechen. Dies kann zum Beispiel in den Fällen sinnvoll sein, in denen Programme oder Programmfolgen »aus dem Ruder laufen«, zum Beispiel in Folge eines Bedienungs- oder Programmierfehlers.

Zum Abbrechen von Programmen dienen eine Reihe von sogenannten Control-Kombinationen, bei denen Sie die [Ctrl]-Taste festhalten und dann eine andere Taste drücken müssen.

So bricht zum Beispiel [Ctrl]-[C] das gerade laufende Programm ab und [Ctrl]-[D] bricht eine Kommandofolge vor dem nächsten Befehl ab, läßt den gerade aktiven Befehl aber zu Ende arbeiten. Kommandofolgen werden mit dem *execute*-Befehl gestartet, der in einem folgenden Kapitel noch näher erläutert wird. Es gibt auch noch die Abbruchsbefehle [Ctrl]-[E] und [Ctrl]-[F], die aber so gut wie nie verwendet werden. Deshalb werden sie nicht weiter behandelt. Probieren Sie zunächst einmal [Ctrl]-[C] aus. Verwenden Sie dazu den Ihnen inzwischen ja gut bekannten *dir*-Befehl. Geben Sie wieder die folgende Zeile ein (und schließen Sie wie gewohnt mit [Return] ab):

```
1>dir
```

Halten Sie nun die [Ctrl]-Taste fest und warten Sie, bis einige Zeilen am Bildschirm erschienen sind. Drücken Sie dann [C] (während Sie die [Ctrl]-Taste weiter festhalten). Der dir-Befehl wird dann abgebrochen und es erscheint die folgende Meldung auf dem Bildschirm (die soviel wie »Unterbrechung!« besagt):

```
*** BREAK
```

Zu diesen Abbruchmöglichkeiten über die Tastatur muß man jedoch sagen, daß kein Programm gezwungen wird, darauf zu reagieren. Alle diese Control-Kombinationen setzen nur eine Art »Signal«, um das sich ein Programm kümmern kann, aber nicht kümmern muß. Gut geschriebene Programme prüfen diese Signale aber regelmäßig und halten sich auch daran.

7.4 Dateien und Directories

Mit dem Grundwissen über das CLI, das Sie in den ersten Abschnitten dieses Kapitels erworben haben, sind Sie schon recht gut gerüstet für die Arbeit mit dem CLI. Einige wichtige Konzepte von AmigaDOS (das dem CLI ja zugrunde liegt) müssen aber noch erwähnt werden, bevor Sie mit der Arbeit beginnen können. Die wichtigsten dieser Konzepte sind wohl Dateien, Dateiverzeichnisse und Geräte. Wer schon andere Computer kennt, dem werden diese Begriffe vielleicht schon vertraut sein. Wenn Sie bisher aber nur die Workbench des Amiga kennen, soll Ihnen der folgende Abschnitt einen ungefähren Eindruck von diesen Begriffen aus der Sprache »Computerchinesisch« geben.

7.4.1 Was eine Datei ist

In diesem Kapitel (und auch schon in früheren Kapiteln) ist häufig das Wort »Datei« gefallen. Was ist denn nun eine solche Datei? Dateien haben Sie bereits im ersten Teil dieses Buches kennengelernt, der sich mit der Workbench beschäftigte. Programme (Werkzeuge) und Dokumente (Projekte) sind zum Beispiel Dateien. Während eine Bezeichnung wie »Programm« aber schon etwas über die Aufgabe des entsprechenden Objekts sagt, ist »Datei« etwas abstrakter. Eine Datei ist einfach eine Ansammlung von Daten, die zu einem zusammen-

hängenden Objekt, eben einer Datei, zusammengefaßt wurden. Ein solches Objekt kann mit einem Namen bezeichnet werden, dem »Dateinamen«.

Die Disketten oder Festplatten des Amiga sind zunächst einmal auch eine Folge von Daten (Bytes), die aus technischen Erwägungen zu größeren *Blöcken* zusammengefaßt werden. Diese Blöcke haben fortlaufende Nummern. Gäbe es keine Dateien, so müßte man jedesmal, wenn man Daten von einer auf eine ander Diskette speichern will, angeben, in welchem Block die Daten liegen und wo sie abgelegt werden sollen. Wenn die Daten länger sind als ein Block, so müssen sie sich auf einen zweiten, dritten, vierten, und so weiter Block erstrecken. In welchen Blöcken diese zusammengehörigen Daten liegen, muß man sich natürlich auch merken. Ein Kopierbefehl könnte dann lauten: »Nimm die 2 Byte von dieser Stelle der Diskette, die 374 Byte von dort und die 1235 Byte von hier und lege Sie da und dort auf die andere Diskette.« Stattdessen kann man eine solche Ansammlung von Daten (Datei) zum Beispiel »Brief« nennen. Was diese Daten sind oder wie sie strukturiert sind, kümmert das Betriebssystem des Amiga, Amiga-DOS, dabei nicht. Egal, ob es sich um ein Programm oder um einen Brief an einen Geschäftsfreund handelt, Amiga-DOS sieht nur eine Folge von Bytes, die irgendwo auf der Diskette oder einer Festplatte abgespeichert sind und einen eigenen Namen haben.

Das Konzept der Datei enthebt uns (beziehungsweise ein Programm) von diesen Bit- und Byte-Manipulationen und allem »unwichtigen« Verwaltungskram. Für Programme ist eine Datei eine Folge von Bytes (Buchstaben), die einen eindeutigen Namen besitzt. An diese Folge kann man hinten weitere Daten anhängen, wahlweise ein bestimmtes Byte aus der Folge lesen, oder auch modifizieren, und so weiter. Das Betriebssystem des Amiga, Amiga-DOS, sorgt für die Zuordnung dieser Bytes in Blöcke auf der Diskette oder Festplatte. Es übernimmt den Verwaltungskram, wie wir auch von einer guten Sekretärin oder einem Sekretär erwarten, daß sie alle einzelnen Blätter, Formulare und so weiter aus den verschiedenen Schränken eines Büros hervorsuchen, wenn wir die *Akte Meier* anfordern.

7.4.2 Was ein Directory ist

Wenn Dateien einzelnen Akten entsprechen, so sind Dateiverzeichnisse die Schubladen und Aktenordner unseres elektronischen Büros. Sie haben diese Aktenordner bereits kennengelernt: Was die Workbench als Schubladen zeigt, sind nahezu die gleichen Objekte, die AmigaDOS und das CLI als »Directory« (*Inhaltsverzeichnis* oder *Dateiverzeichnis*) bezeichnet.

Ein Directory ist eine Liste von Dateinamen und Informationen, wo diese Dateien auf der Diskette zu finden sind. Weiterhin sind darin auch zusätzliche Informationen über die einzelnen Dateien, wie Größe, Datum der letzten Änderung und zusätzliche Kommentare, verzeichnet. Diese (auf der Diskette abgespeicherte) Liste entspricht ziemlich genau der Liste, die am Bildschirm auftaucht, wenn wir im CLI den Befehl *dir(*ectory) eingeben.

7.4.3 Der Dateienbaum

Werden unsere Aktenmengen noch größer, so können die Aktenberge auch trotz Schubladen beziehungsweise Directories unübersichtlich werden. Wieso sollten wir dann nicht auch die Schubladen wieder gliedern? Genau das ist im CLI genauso wie auf der Workbench möglich. Dateiverzeichnisse können in Dateiverzeichnissen liegen und selbst natürlich auch wieder Dateiverzeichnisse enthalten.

Ein anderer möglicher Vergleich wäre ein Buch: Wie ein gutes Buch eine vielstufige Gliederung aus Buchteilen, Kapiteln, Abschnitten, Unterabschnitten und so weiter hat, kann auch eine gut aufgeräumte Diskette, und vor allem eine ja noch viel größere Festplatte eine vielstufige Hierarchie von Dateien und Dateiverzeichnissen haben, die es uns erleichtert, unter den vielen Dateien eine bestimmte zu finden.

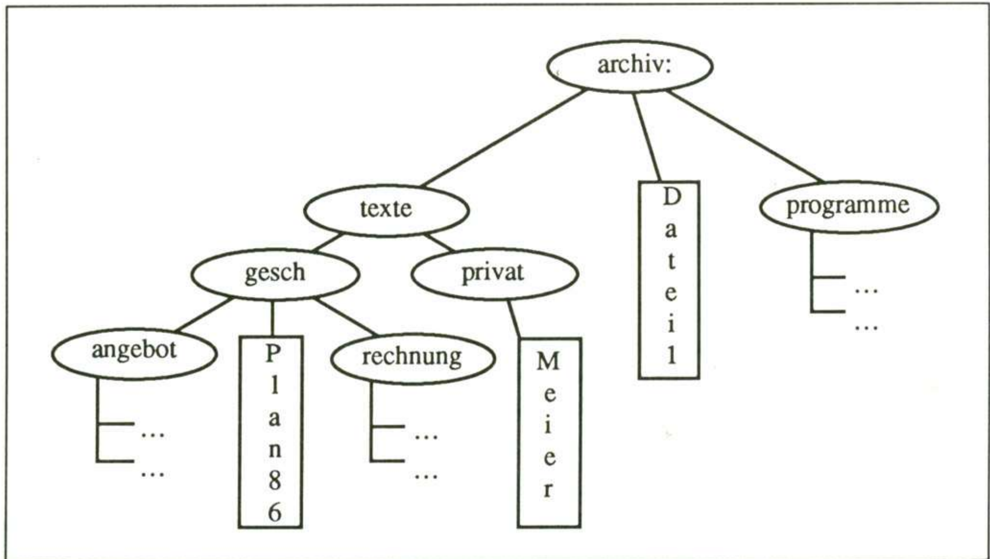


Bild 7.4: Ein Dateienbaum auf einer Diskette

Die obige Abbildung zeigt einen solchen Dateienbaum, wie er auf einer Diskette existieren könnte. Dateiverzeichnisse sind in diesem Baum durch Ovale, Dateien durch Rechtecke gekennzeichnet. (Einige Zweige des Baums sind aus Platzgründen nur durch Pünktchen angedeutet.) Es entspricht einer allgemein gültigen Konvention, solche Dateienbäume »auf dem Kopf«, also mit der Wurzel obenliegend darzustellen. Lassen Sie sich dadurch nicht verwirren – es ist einfach so üblich.

Eine solche Hierarchie darf aber auch nicht zu groß und unübersichtlich werden. Es kann sonst extrem schwierig werden, eine Datei genau zu bestimmen. Denn anders als auf der Workbench

ist es im CLI nicht möglich, auf eine Datei (ein Piktogramm) zu zeigen. Man muß statt dessen, wenn man eine Datei manipulieren (zum Beispiel löschen) will, genau die Folge der Schubladen (Directories) angeben, die man nacheinander öffnen muß, um zu ihr zu gelangen. Diese Aneinanderreihung von Directory- und Dateinamen ist der sogenannte Pfadname einer Datei, weil er beschreibt, auf welchem Pfad man zu der Datei gelangt.

7.4.4 Pfadnamen – Wege zu Dateien

Jede Datei und jedes Directory auf einer Diskette hat einen ganz bestimmten (vollständigen) Pfadnamen. Er beginnt mit der »Wurzel« des Dateienbaums. Diese Wurzel hat keinen eigenen Namen, sondern trägt den Namen der Diskette, wie er auf der Workbench unter dem Disketten-Piktogramm auftauchen würde. Heißt die Diskette zum Beispiel *Archiv*, so wäre der Name der Wurzel auf dieser Diskette *Archiv*. Es spielt dabei keine Rolle, in welchem Laufwerk sich die Diskette befindet! Der vollständige Pfadname einer Datei beginnt immer mit dem Namen der Wurzel (um Dateien auf verschiedenen Disketten auseinanderhalten zu können) gefolgt von einem Doppelpunkt. Dann kommen die Schubladen, die man nacheinander öffnen muß, um zu dieser Datei zu gelangen, getrennt von Schrägstrichen. Erst als letzter Bestandteil des Pfadnamens kommt der Dateiname selbst.

Die Datei *Meier* in der Schublade *Privat*, die in der Schublade *Texte* auf der Diskette *Archiv* liegt, hätte demnach also den vollständigen Pfadnamen *Archiv:Texte/Privat/Meier* und die Datei Müller in derselben Schublade den Pfadnamen *Archiv:Texte/Privat/Müller* (der erste Teil bleibt gleich).

7.4.5 Das aktuelle Directory

In dem obigen Beispiel mit den nur zweifach verschachtelten Directories *Privat* und *Texte* wurde der Pfadname schon recht lang. Wenn Sie auf Ihrer Diskette die Directories (Schubladen) aber noch tiefer ineinandergeschachtelt oder lange Namen verwendet haben, kann der vollständige Pfadname rasch 30, 40 oder noch mehr Buchstaben lang werden. Das ist nicht nur lästig, sondern erhöht auch die Gefahr von Tippfehlern enorm.

Amiga-DOS und das CLI kennen deshalb das Konzept des »aktuellen Directory«. Ein Directory aller Directories auf allen Disketten und Festplatte, die im Moment angeschlossen sind, ist das aktuelle Directory. Um eine Datei im aktuellen Directory zu benennen, braucht man nicht den Pfadnamen zu verwenden, sondern es reicht, den Dateinamen anzugeben. Die folgende Abbildung zeigt wieder einen Dateienbaum. In diesem ist das aktuelle Dateiverzeichnis durch ein fettes Oval markiert und zwei Dateien sind durch ein fettes Rechteck hervorgehoben.

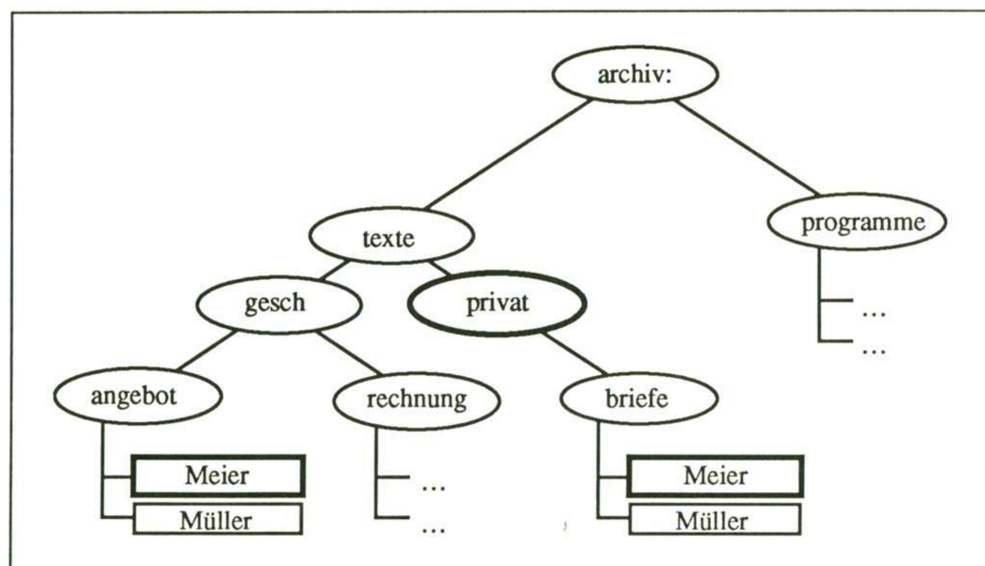


***Bild 7.5:** Pfadnamen im Dateienbaum*

Die Namen, die man für diese Dateien im CLI eintippen müßte, würden *archiv:texte/gesch/angebot/Meier* und *briefe/Meier* lauten (beim aktuellen Dateiverzeichnis *archiv:texte/privat*. Wäre gar *archiv:texte/privat/briefe* das aktuelle Directory, würde es für die rechte der beiden Dateien genügen, einfach *Meier* zu sagen.

Sie sehen schon an diesen kleinen Beispielen, wieviel Schreibarbeit Sie sich auf diese Weise ersparen können. In einem der folgenden Abschnitte werden Sie deshalb den Befehl *cd* (*change directory = ändere Dateiverzeichnis*) kennenlernen, mit dem Sie jederzeit ein bestimmtes Directory zum aktuellen Directory »ernennen« können.

7.4.6 Die aktuelle Diskette

Ähnlich, wie es immer ein aktuelles Directory gibt, existiert auch immer eine »aktuelle Diskette«. Dies ist die Diskette, auf der sich das aktuelle Directory befindet. Wenn Sie also (mit dem Befehl *cd*) das aktuelle Directory wechseln und das neue aktuelle Directory sich auf einer anderen Diskette befindet, wird diese dadurch gleichzeitig zur aktuellen Diskette.

Man kann die aktuelle Diskette - ähnlich wie das aktuelle Directory – zur Arbeitserleichterung verwenden. Wenn sich eine Datei, deren Pfadnamen Sie eingeben wollen, nämlich auf der aktuellen Diskette befindet, können Sie den Diskettennamen im Pfadnamen fortfallen lassen (nicht jedoch den Doppelpunkt hinter dem Diskettennamen). Statt *archiv:texte/privat/Meier* könnten Sie deshalb auch *:texte/privat/Meier* schreiben, um die Datei *Meier* in der Schublade *privat* zu bezeichnen.

7.4.7 Laufwerksnamen

Wie Sie gesehen haben, enthält der vollständige Pfadname einer Datei immer auch den Namen der Diskette, auf der sich die Datei befindet. Dadurch wird praktisch die Wurzel des Dateienbaums festgelegt, in dem die Datei zu finden ist. Auf jeder Diskette befindet sich nämlich ein eigener Dateienbaum, der mit dem einer anderen Diskette keine Verbindung (auch keine gemeinsame Wurzel) hat.

Als Diskettenname wurde bislang derselbe Name verwendet, der auch auf der Workbench unter dem Piktogramm der Diskette steht. Die Verwendung dieses Namens kann aber manchmal unpraktisch sein (weil er zu lang ist) und manchmal schwierig sein (weil man ihn vergessen hat und ihn auch nicht sieht, wenn das CLI-Fenster das Disketten-Piktogramm verdeckt). In solchen Fällen ist es ganz praktisch, daß es noch eine zweite Möglichkeit gibt, eine Diskette zu bezeichnen. Und zwar ist dies der Name des Diskettenlaufwerks, in dem sich die Diskette befindet. (Ja Sie haben richtig gelesen; auch Diskettenlaufwerke haben Namen – allerdings merkwürdige.) Das interne Laufwerk trägt den Namen *DF0*, das erste externe heißt *DF1* und falls Sie weitere externe Laufwerke anschließen, haben diese die Namen *DF2* und *DF3* (mehr als vier Diskettenlaufwerke können Sie nicht verwenden.)

Sie können also die Datei *archiv:texte/privat/Meier* auch mit *df0:Texte/Privat/Meier* ansprechen, wenn sich die Diskette *archiv* im internen Laufwerk befindet. Beachten Sie aber, daß diese Datei auf einmal *df1:texte/privat/Meier* heißt, wenn die entsprechende Diskette im externen Laufwerk liegt. Das kann verwirrend sein. Trotzdem sind diese Laufwerksnamen ganz praktisch, da man ja immer weiß, in welchem Laufwerk sich eine Diskette gerade befindet, auch wenn man sich an ihren Namen gerade nicht erinnern kann.

7.5 Arbeiten mit dem CLI

Nachdem Sie nun – zumindest ungefähr – wissen, was es mit Dateien und Directories auf sich hat, können Sie in diesem Abschnitt Ihre neugewonnenen Kenntnisse ausprobieren.

7.5.1 Wie Sie Directories betrachten

Den Befehl, mit dem Sie ein Directory betrachten können, haben Sie ja bereits kennengelernt: *dir*. Wenn Sie im CLI *dir* eintippen und dann die [Return]-Taste drücken, bekommen Sie den Inhalt des aktuellen Directories angezeigt. Nach dem Aufruf des CLI ist dies das Wurzelverzeichnis auf der Workbench-Diskette, das in etwa denselben Inhalt hat wie das Diskettenfenster der Workbench-Diskette. Probieren Sie den *dir*-Befehl einfach einmal aus. Machen Sie dazu aber das CLI-Fenster so groß wie möglich.

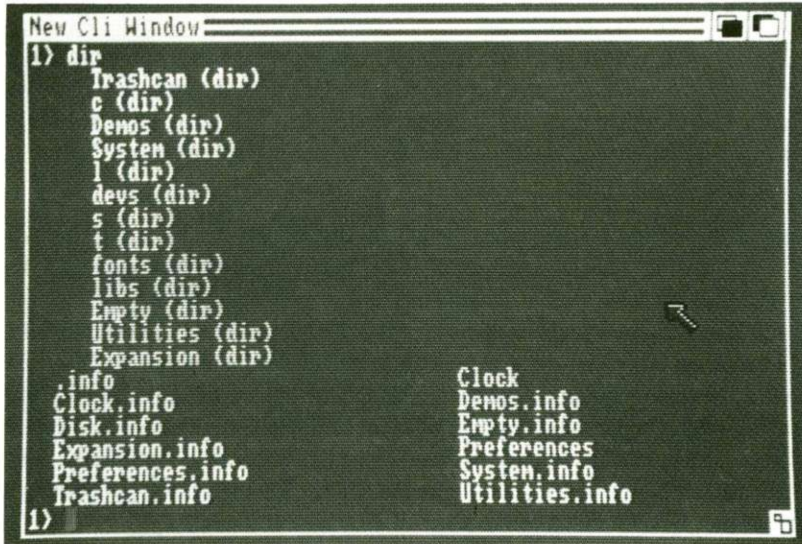


Bild 7.6: Auflistung eines Directory

In der Liste, die der *dir*-Befehl ausgibt, kommen zwei verschiedene Arten von Namen vor. Die eine Sorte Namen wird einspaltig aufgelistet und jedem Namen folgt in Klammern das Wort *dir*. Dies sind andere Directories, die im Wurzelverzeichnis legen. Einen Teil davon können Sie auch als Schubladen im Diskettenfenster der Workbench-Diskette sehen. Die andere Sorte Namen wird zweispaltig aufgelistet. Dies sind die Namen der Dateien, die sich im Wurzelverzeichnis der Diskette befinden. Auch einige dieser Namen tauchen als Piktogramme auf, andere nicht. Wenn Sie die Namensliste einmal näher betrachten, wird Ihnen auffallen, daß viele Namen in ähnlicher Form zweimal vorkommen. So gibt es zum Beispiel *Preferences* und *Preferences.info*. Betrachten Sie dann die Liste genauer, wird Ihnen auffallen, daß genau die Namen noch einmal mit einem .info dahinter auftauchen, die auch als Piktogramme auf der Workbench sichtbar sind. Das kommt nicht von ungefähr, wie Sie gegen Ende dieses Kapitels noch erfahren werden.

Sie können mit dem *dir*-Befehl aber nicht nur das aktuelle, sondern auch andere Directories betrachten. Dazu müssen Sie hinter dem Wort *dir* nur eine oder mehrere Leerstellen eingeben und dann den Pfadnamen des Directory eintippen, dessen Inhalt Sie sehen wollen, bevor Sie auf [Return] drücken. Geben Sie zum Beispiel einmal die folgende Zeile ein:

```
1>dir  c
```

Sie bekommen daraufhin den Inhalt des Directory *c* auf der Workbench gezeigt. Stoppen Sie die Auflistung zwischendurch ruhig einmal mit einem Druck auf die Leertaste (Fortsetzen mit [Backspace]), um sich die Liste in Ruhe anzuschauen. Diese Liste enthält fast alle Befehle (engl. *commands*), die das CLI »versteht«. Im Directory *c* befinden sich nämlich die

Programme, die aufgerufen werden, wenn Sie einen Befehl im CLI eingeben. Wenn Sie zum Beispiel *dir* eintippen, wird daraufhin das Programm *dir* im Dateiverzeichnis *c* ausgeführt.

Probieren Sie ruhig ein wenig die Möglichkeiten des *dir*-Befehls aus. Schauen Sie auch die Inhalte anderer Dateiverzeichnisse an. Deren Namen sehen Sie ja, wenn Sie den Befehl *dir* ganz ohne Parameter eingeben. Es sind die Namen, hinter denen in dieser Liste »(dir)« steht.

Der Parameter von *dir* muß übrigens unbedingt ein Directory sein. Wenn Sie *dir* den Namen einer Datei (zum Beispiel eines Programmes) als Parameter übergeben, erhalten Sie die folgende Fehlermeldung (die soviel heißt wie »Dateiname ist kein Dateiverzeichnis«):

```
1>dir preferences
filename is not a directory
```

Der Befehl *dir* zeigt aber nur die Namen der Dateien und Directories. Es gibt noch einen anderen Befehl, der Ihnen mehr Informationen über die einzelnen Objekte in einem Directory mitteilt. Dieser Befehl heißt *list*. Geben Sie die folgende Zeile im CLI-Fenster ein, um *list* auszuprobieren:

```
1>list devs/printers
```

Wenn Sie *list* ohne einen Parameter dahinter verwenden, wird wie beim *dir*-Befehl der Inhalt des aktuellen Directory aufgelistet. Mit *list devs/printers* erhalten Sie eine Auflistung des Inhalts des Directory *printers* im Directory *devs* auf der Workbench-Diskette. In jeder Zeile erhalten Sie Informationen über eine Datei oder ein Directory. Nebeneinander werden der Dateiname, die Dateigröße (in Bytes), die »Schutz-Flags« und Uhrzeit und Datum beim Erzeugen dieser Datei angezeigt. Unter Umständen wird in der Zeile unter dem Namen auch noch ein Kommentar angezeigt (die meisten Dateien haben aber keinen solchen Kommentar).

Der Befehl *list* liefert also viel mehr Informationen über jede Datei als *dir*. Diese Informationen brauchen aber dafür auch länger, um über den Bildschirm zu laufen. Es ist deshalb meist einfacher und geht schneller, *dir* zu verwenden, wenn man nur »schnell mal eben« eine bestimmte Datei sucht.

7.5.2 Wie Sie Programme starten

Genausoleicht wie das Erteilen eines Befehls (zum Beispiel *dir*) ist es, ein Programm (Werkzeug) vom CLI aus zu starten. Genaugenommen, haben Sie das ja auch schon getan, denn auch *dir* und *list*, die beiden Befehle, die Sie schon kennengelernt haben, sind ja Programme. Um ein Programm zu starten, brauchen Sie nur seinen Namen beziehungsweise Pfadnamen einzugeben und die [Return]-Taste zu drücken. Probieren Sie es aus: Geben Sie die folgende Zeile ein und drücken Sie [Return].

```
1>:preferences
```

Nach wenigen Augenblicken müßte daraufhin das Hauptfenster von *Preferences* erscheinen. Sie haben auf diese Weise *Preferences* ohne Berührung der Maus vom CLI aus gestartet. (Falls Sie statt dessen eine Fehlermeldung erhalten, haben Sie sich vielleicht vertippt. Versuchen Sie

es dann noch einmal.) Jetzt brauchen Sie allerdings die Maus, um das Programm wieder verlassen zu können. Klicken Sie auf den Knopf Cancel unten rechts im Fenster.

Sie können auf diese Weise jedes beliebige Programm starten, wenn Sie seinen Pfadnamen kennen. Zu diesem Thema allerdings noch eine kleine Warnung: Es gibt einige Programme aus der Frühzeit des Amiga 1000, die vom CLI aus nicht gestartet werden können. Ein Versuch, sie vom CLI aus zu starten, ergibt eine Fehlermeldung oder einen »Absturz« des Amiga 500 (bei dem ein Alert erscheint).

7.5.3 Wie Sie Disketten kopieren

Einer der ersten »Handgriffe« die Sie auf der Worbench gelernt haben, war das Kopieren einer Diskette. Dies ist eine sehr wichtige Arbeit, die Sie regelmäßig mit allen wichtigen Disketten machen sollten. Selbstverständlich gibt es auch im CLI einen Befehl zu diesem Zweck, der sinnigerweise *diskcopy* heißt. Er kann sowohl mit einem als auch mit zwei Laufwerken dazu verwendet werden, nahezu identische Kopien kompletter Disketten anzulegen. Legen Sie sich dazu jetzt eine leere, neue Diskette oder eine bereits gebrauchte Diskette, deren Inhalt Sie nicht mehr benötigen, bereit und aktivieren Sie sicherheitshalber den Schreibschutz der Diskette, mit der Sie gerade arbeiten. Geben Sie dann die folgende Zeile ein:

```
1>diskcopy FROM df0: TO df0: NAME "CLI 2"
```

Das bedeutet, daß Sie vom (engl. *from*) internen zum (engl. *to*) internen Laufwerk kopieren wollen und die kopierte Diskette den Namen (engl. *name*) »CLI 2« erhalten soll. Dabei sind »df0:« und »"CLI 2"« die Parameter des Befehls *diskcopy* (wie der Directoryname, der *dir* folgen kann) und FROM, TO und NAME sind Schlüsselwörter, die dem Befehl mitteilen, wo welcher Parameter steht. Nebenbei dienen solche Schlüsselwörter auch noch dazu, einen CLI-Befehl lesbarer zu gestalten – zumindest für den, der der englischen Sprache mächtig ist. Wenn Sie diese Zeile eingegeben und auf [Return] gedrückt haben, erscheint die folgende Meldung im CLI-Fenster:

```
Place SOURCE disk ( FROM disk ) in drive DF0:
Press RETURN to continue
```

Das heißt soviel wie »Legen Sie die Originaldiskette in das interne Laufwerk und drücken Sie die [Return]-Taste um fortzufahren«. Legen Sie daraufhin (falls nötig) die Diskette, die Sie kopieren wollen, in das interne Laufwerk Ihres Amiga 500 und drücken Sie die [Return]-Taste. Der Inhalt dieser Diskette wird nun in den RAM-Speicher Ihres Amiga 500 eingelesen. Dieser Vorgang wird auch am Bildschirm protokolliert:

```
Reading 9, 70 to go
```

Das bedeutet, daß bereits <nn> »Spuren« der Diskette eingelesen wurden. Eine Diskette besitzt insgesamt 80 Spuren, so daß Sie an dieser Zahl ungefähr ablesen können, »wieviel schon geschafft ist«. Ist der RAM-Speicher voll, erscheint die folgende Meldung:

```
Place DESTINATIPON disk ( TO disk ) in drive DF0:
Press RETURN to continue
```

Das heißt soviel wie »Legen Sie die Zieldiskette (die die Kopie aufnehmen soll) in das interne Laufwerk und drücken Sie die [Return]-Taste um fortzufahren«. Legen Sie daraufhin die bereitgelegte leere Diskette in das interne Laufwerk Ihres Amiga 500 und drücken Sie die [Return]-Taste. Der Inhalt der Original-Diskette, der in den RAM-Speicher eingelesen worden ist, wird nun auf diese Diskette geschrieben. Auch das wird wieder am Bildschirm protokolliert:

```
WRITING 12, 67 to go
```

Das bedeutet, daß bereits <nn> »Spuren« der Diskette geschrieben wurden. Wenn der Amiga 500 damit fertig ist, fordert er wieder die Originaldiskette an und liest das nächste »Häppchen« ein. Dann fordert er die Zieldiskette an und schreibt dieses Häppchen darauf und so weiter und so fort. Wieviele Diskettenwechsel nötig sind, hängt davon ab, wieviel Speicher Ihr Amiga 500 besitzt, welche Programme Sie vorher gestartet haben, wieviele Fenster offen sind und so weiter. Es ist also nicht exakt vorhersehbar. Wenn der Kopiervorgang beendet ist, erscheint die folgende Meldung:

```
Diskcopy Finished
```

Falls während des Kopiervorgangs ein Fehler gemeldet wird (zum Beispiel »Disk Error« oder eine ähnliche Meldung), ist eine der beiden Disketten defekt. Falls es die Zieldiskette (*Target Disk*) ist, sollten Sie diese in den Müll werfen – oder besser zum Händler zurückbringen. Ist es die Originaldiskette, sollten Sie möglichst schnell alle wichtigen Dateien von dieser Diskette auf eine andere Diskette kopieren. Gehen Sie dazu am besten in die Workbench und legen Sie ein Piktogramm nach dem anderen auf die andere Diskette. Wenn dabei erneut ein Fehler auftaucht, müssen Sie auf das Piktogramm verzichten, bei dem dieser Fehler aufgetaucht ist. Sie können danach die defekte Diskette mit *Initialize* aus dem *Disk*-Menü löschen und initialisieren. Wenn dabei wieder ein Fehler auftaucht, werfen Sie am besten diese Diskette in den Müll oder bringen Sie dem Händler zurück, der sie Ihnen verkauft hat.

Aber auch abgesehen von diesen Fehlermöglichkeiten, ist das Kopieren mit einem Laufwerk schon mühselig genug. Einfacher ist es mit zwei Laufwerken. Wenn Sie ein zweites Laufwerk besitzen, können Sie das ausprobieren. Geben Sie dazu die folgende Zeile ein und beenden sie wie üblich mit [Return]:

```
1>diskcopy FROM df0: TO df1: NAME "CLI 2"
```

Das bedeutet, daß Sie vom (engl. *from*) internen zum (engl. *to*) externen Laufwerk *DF1:* kopieren wollen und die kopierte Diskette den Namen (engl. *name*) »CLI 2« erhalten soll. Daraufhin erscheint die folgende Meldung im CLI-Fenster:

```
Place SOURCE disk ( FROM disk ) in drive DF0:
Place DESTINATIPON disk ( TO disk ) in drive DF1:
Press RETURN to continue
```

Das heißt soviel wie »Legen Sie die Originaldiskette in das interne Laufwerk, die Diskette, die die Kopie aufnehmen soll, in das externe Laufwerk und drücken Sie die [Return]-Taste um fortzufahren«. Befolgen Sie diese Anweisungen. Der Inhalt der Diskette im internen Laufwerk

wird nun direkt auf die Diskette im externen Laufwerk geschrieben. Das mühselige Austauschen der Disketten entfällt. Wenn der Kopiervorgang abgeschlossen ist, erscheint wieder die folgende Meldung im CLI-Fenster:

```
Diskcopy Finished
```

7.5.4 Wie Sie eine Datei kopieren

Der *diskcopy*-Befehl dürfte Ihnen schon mal einen ersten Eindruck davon vermittelt haben, wie ein Befehl mit mehreren Parametern aussieht. Die im folgenden noch beschriebenen Befehle sind meist einfacher aufgebaut. Wenn Sie zum Beispiel eine Datei kopieren wollen, so benötigen Sie dazu den – sehr leicht zu bedienenden – *copy*-Befehl (engl. *to copy = kopieren*).

Das folgende Beispiel geht davon aus, daß Sie mit einer Kopie der Original-Workbench-Diskette arbeiten und auf dieser zumindest noch ein wenig Platz ist. Geben Sie folgendes ein (und achten Sie dabei auf eventuelle Tippfehler):

```
1>copy FROM disk.info TO test1
```

Das Diskettenlaufwerk beginnt nun einen Moment lang zu arbeiten. Der *copy*-Befehl legt eine Kopie der Datei *disk.info* in der Datei *test1* an. (Überzeugen Sie sich mit Hilfe des *dir*-Befehls davon.) *Copy* benötigt zwei Parameter, den Namen der Datei, die kopiert werden soll (hinter dem Schlüsselwort FROM) und den Namen, den die Kopie tragen soll (hinter dem Schlüsselwort TO). Das Schlüsselwort FROM ist dabei »optional«, das heißt, Sie brauchen es nicht unbedingt vor den Namen der zu kopierenden Datei schreiben. Der folgende Befehl erzeugt zum Beispiel eine zweite neue Datei mit dem Namen *test2*.

```
1>copy test1 TO test2
```

Wenn Sie die Reihenfolge der beiden Parameter, so wie bisher beschrieben, einhalten (zuerst den Namen der Originaldatei, dann den der Kopie), brauchen Sie auch das Schlüsselwort TO nicht zu verwenden. Der folgende Befehl erzeugt zum Beispiel eine dritte neue Datei mit dem Namen *test3*. (Überzeugen Sie sich davon mit Hilfe des *dir*-Befehls.)

```
1>copy test1 test3
```

Wie Sie sehen, ist das Kopieren einer Datei sehr einfach. Man braucht sich eigentlich nur den Namen des Befehls *copy* dafür zu merken!

Wenn Sie sich bei den oben genannten Beispielen einmal vertippt haben, werden Sie merken, daß das CLI sehr pingelig ist. Ein kleiner Tippfehler führt meist dazu, daß das Amig-Dos entweder nicht das Richtige tut oder daß der Befehl gar nicht ausgeführt wird und nur eine Fehlermeldung erscheint. In anderer Hinsicht ist das CLI eher großzügig. So spielt die Groß- beziehungsweise Kleinschreibung in Befehlen keine Rolle. Sowohl in Befehlsnamen wie auch in Dateinamen und Schlüsselwörtern können Sie Groß- und Kleinbuchstaben nach Herzenslust mischen, ohne daß dadurch die Bedeutung des Befehls verändert würde. Der folgende Befehl leistet deshalb exakt dasselbe wie das letzte Beispiel:

```
1>cOPy TEs2 tO test3
```

7.5.5 Wie Sie eine Datei umbenennen

Genauso einfach ist es auch, eine Datei umzubenennen. Der entsprechende Befehl heißt, genau wie auf der Workbench, *rename*. Probieren Sie ihn am besten gleich einmal aus und geben Sie die folgende Zeile im CLI-Fenster ein (Sie müssen dazu aber die Beispiele im vorigen Abschnitt ausprobiert und dabei die drei Dateien *test1*, *test2* und *test3* erzeugt haben):

```
1>rename test3 TO testNEU
```

Wenn Sie danach den *dir*-Befehl aufrufen, werden Sie sehen, daß es danach die Datei *test3* nicht mehr gibt, dafür aber die Datei *testNEU*. Das ist natürlich *test3*; nur unter einem anderen Namen. Statt des Schlüsselwortes TO können Sie bei *rename* übrigens auch das Schlüsselwort AS (engl. für *als*) verwenden oder auch ganz ohne Schlüselwort arbeiten wie beim *copy*-Befehl. Wenn Sie die folgenden beiden Zeilen nacheinander eingeben, sollten Sie wieder drei Dateien namens *test1*, *test2* und *test3* sehen, wenn Sie sich das Wurzelverzeichnis mit dem *dir*-Befehl anschauen:

```
1>rename testNEU AS xxx
1>rename xxx test3
```

Versuchen Sie nun auch einmal, eine Datei umzubenennen und ihr dabei denselben Namen zu geben, den eine schon existierende Datei besitzt:

```
1>rename test3 TO test2
```

Sie erhalten daraufhin nur die folgende Fehlermeldung:

```
Can't rename test3 as test2
```

Sie besagt nichts anderes, als daß es unmöglich ist, einer Datei mit *rename* den Namen einer schon existierenden Datei zu geben. Der Befehl wird nicht ausgeführt.

7.5.6 Wie Sie eine Datei löschen

Nachdem Sie nun gesehen haben, wie man neue Dateien erzeugt (durch Kopieren schon vorhandener Dateien) und dann wieder umbenennt, werden Sie jetzt erfahren, wie Sie diese Dateien auch wieder loswerden. Für solche Aufgaben ist der *delete*-Befehl zuständig (engl. *to delete = löschen*). Geben Sie die folgende Zeile im CLI-Fenster ein und überzeugen Sie sich dann mit dem *dir*-Befehl davon, daß die Datei *test3* nun fehlt:

```
1>delete test3
```

Wie Sie sehen, brauchen Sie sich beim *delete*-Befehl nicht einmal ein Schlüsselwort zu merken. Sie können mit diesem Befehl auch mehr als nur eine Datei löschen. Hierzu brauchen Sie nur die Namen der Dateien, die Sie löschen wollen, alle hinter *delete* zu schreiben und [Return] zu drücken. Sie müssen dazu allerdings die Namen in dieser Liste jeweils mit einer

Leerstelle voneinander trennen. Probieren Sie es aus: Erzeugen Sie (mit *copy*) aus *test2* die beiden neuen Dateien *test3* und *test4* und löschen Sie diese dann mit dem folgenden Befehl

```
1>delete test3 test4
```

Überzeugen Sie sich vor und nach diesem Befehl mit Hilfe des *dir*-Befehls davon, daß die beiden neuen Dateien *test3* und *test4* auch wirklich vorhanden sind beziehungsweise von *delete* gelöscht wurden.

7.5.7 Wie Sie den Inhalt einer Datei auf dem Bildschirm betrachten

Wenn Sie Dateien nicht nur hin und her kopieren, umbenennen und wieder löschen wollen, sondern auch gerne wüßten, was diese Datei enthält, kann Ihnen manchmal ein anderer CLI-Befehl helfen. Der Befehl *type* benötigt nur einen einzigen Parameter, einen Dateinamen, und gibt den Inhalt dieser Datei auf dem Bildschirm aus. Wichtig ist, daß es sich bei dieser Datei um eine reine »Textdatei« handelt, die nur Buchstaben enthält. Viele Programme erzeugen Dateien, die nicht nur Text, sondern auch Bilder und Formatierungsinformationen enthalten. Auch Programme selbst sind Dateien, die viel mehr als nur reine Texte enthalten. Wenn Sie versuchen, mit *type* ein Programm oder ein Bild, das Sie in einem Malprogramm gezeichnet haben, auszugeben, erscheint nur Chaos auf dem Bildschirm, dem Sie nichts entnehmen können. Auf der Workbench-Diskette befindet sich aber glücklicherweise schon eine Textdatei, anhand der Sie *type* ausprobieren können. Geben Sie dazu den folgenden Befehl ein (und achten Sie sehr genau auf Tippfehler):

```
1>type s/startup-sequence
```

Daraufhin läuft ein langer Text über den Bildschirm. Es handelt sich dabei um eine recht komplizierte Folge von CLI-Befehlen, die bei jedem Neustart des Amiga 500 ausgeführt wird und über die Sie im übernächsten Kapitel noch mehr erfahren werden.

Ein abschließender Tip zum *type*-Befehl: Vielleicht passiert es Ihnen einmal aus Versehen, daß Sie eine Datei mit *type* auf den Bildschirm ausgeben und dabei merkwürdige Zeichen erscheinen. Es kann dann sogar vorkommen, daß weitere Befehle, die Sie über die Tastatur eingeben, nicht mehr richtig am Bildschirm erscheinen! In diesem Fall hilft unter Umständen die Tastenkombination [Ctrl]+[O] ([Ctrl] festhalten und [O] tippen) oder [Esc] [C] (erst [Esc] drücken und loslassen, dann [C] drücken).

7.5.8 Wie Sie den DIR-Befehl »interaktiv« verwenden

Sie werden vielleicht staunen, aber viele von den in diesem Abschnitt beschriebenen Operationen können Sie auch mit dem *dir*-Befehl erledigen. Sie können mit ihm zum Beispiel nicht nur ein Directory auflisten, wozu Sie diesen Befehl bislang wahrscheinlich verwendet haben, sondern auch Dateien löschen und sich am Bildschirm betrachten (sofern es Text-Dateien sind). Dazu dient die »interaktive« Option von *dir*. Hierzu müssen Sie am Ende der Zeile mit dem *dir*-Befehl (also direkt hinter *dir* oder hinter dem Namen eines Directory) noch »OPT i« eingeben, bevor Sie [Return] drücken. Probieren Sie es aus:

```
1> dir df0: opt i
```

Sie bekommen nun Zeile für Zeile dieselbe Liste gezeigt, die sonst in einem Rutsch über den Bildschirm huschen würde. *dir* gibt jeweils einen Datei- oder Directory-Namen aus und wartet dann auf Ihre Reaktion. Betätigen Sie daraufhin die [Return]-Taste, erscheint der nächste Name der Liste. Probieren Sie es aus: Drücken Sie zweimal auf die [Return]-Taste.

Wenn der Name eines Directory angezeigt wird, können Sie sich dessen Inhalt anschauen. Sie brauchen dazu nur die Taste [E] zu drücken, wenn dieser Name angezeigt wurde, um in dieses Directory einzusteigen. Bei jeder Betätigung der [Return]-Taste erscheint nun ein neuer Datei- oder Directory-Name aus diesem Directory. Probieren Sie es aus: Drücken Sie so lange auf [Return], bis das Directory *fonts* angezeigt wird. Drücken Sie dann auf [E] und [Return]. Sie können nun mit [Return] dieses Directory durchblättern. Wenn Sie auf [B] und [Return] drücken (oder das Directory zu Ende ist), gelangen Sie wieder zum übergeordneten Directory zurück. In diesem Fall ist es das Wurzeldirectory der Diskette.

Wenn Sie das gefunden oder gesehen haben, was Sie sehen wollten oder gesucht haben, können Sie den interaktiven *dir*-Befehl jederzeit abbrechen, indem Sie die Tastenkombination [Ctrl]-[C] und dann [Return] drücken. Machen Sie das jetzt und geben Sie dann noch einmal die folgende Zeile ein:

```
1> dir df0: opt i
```

Drücken Sie nun so lange auf die [Return]-Taste, bis das Directory *s* erscheint. Jetzt drücken Sie wieder die Tasten [E] und [Return]. *dir* zeigt Ihnen daraufhin die Datei *startup-sequence*. Drücken Sie nun die Tasten [T] und [Return]. Daraufhin erscheint der Inhalt von *startup-sequence* auf dem Bildschirm, als hätten Sie den *type*-Befehl verwendet. Immer wenn *dir* Ihnen eine Textdatei zeigt, können Sie sich mit [T] ihren Inhalt anzeigen lassen. (Sie können einen langen Text dabei wie üblich durch Betätigung der Leertaste stoppen und mit [Backspace] wieder fortsetzen.)

Betätigen Sie nun so lange die [Return]-Taste, bis die Datei *test2* erscheint, die Sie vorhin mit dem *copy*-Befehl erzeugt haben. Geben Sie die drei Buchstaben »del« ein und drücken Sie auf [Return]. Sie haben dadurch mit drei Tastendrücken die Datei *test2* gelöscht (engl. *to* ***delete*** = löschen).

7.5.9 Wie Sie mehrere Dateien mit einem Befehl bearbeiten – Muster

Einige der CLI-Kommandos (zum Beispiel *copy*) bieten Ihnen die Möglichkeit, gleichzeitig mehrere Dateien mit einem Befehl zu bearbeiten. Hierzu müssen Sie statt eines Dateinamens ein *Muster* (engl. *pattern*) angeben. Für jeden Dateinamen auf einer Diskette kann man nun feststellen, ob er in dieses Muster »paßt« oder nicht. Paßt er in das Muster, wird er in eine Liste aufgenommen und mit dem jeweiligen Befehl bearbeitet. Passen zum Beispiel vier Dateinamen in ein bestimmtes Muster und man verwendet dieses Muster bei einem Befehl, so wirkt dieser eine Befehl mit Muster als würde man viermal hintereinander denselben Befehl verwenden und dabei das Muster gegen jeweils einen der vier Dateinamen austauschen.

Bei manchen Befehlen ist es möglich, an jeder Stelle ein Muster zu verwenden, an der normalerweise ein Dateiname stehen muß. Manchmal muß das Muster aber auch durch ein

spezielles Schlüsselwort eingeleitet werden (zum Beispiel beim Befehl *list*). Der Informatiker nennt solche und ähnliche Muster »reguläre Ausdrücke«. Diese Muster sind für die verschiedensten Zwecke in der Informatik recht weit verbreitet. Es gibt andere Betriebssysteme, zum Beispiel UNIX™, die es dem Benutzer erlauben, bei fast jedem Befehl solche Muster für Dateinamen zu verwenden, beim Amiga gibt es jedoch (leider) nur einige Befehle, die diese verstehen.

Ein kurzer Tip vorweg: Für einen Menschen – und Sie sind ja ein Mensch und kein Computer – ist es nicht immer ganz einfach zu durchschauen, welche Dateien in ein solches Muster passen. Dies gilt besonders in der Anfangszeit, solange Sie noch nicht so gut mit Mustern vertraut sind. Gerade bei Befehlen, die Dateien modifizieren oder löschen können, kann es deshalb zu »peinlichen« Effekten kommen, wenn plötzlich mehr oder andere Dateien durch einen Befehl erfaßt werden, als Sie dachten. Solange Sie nicht ganz sicher sind, welche Dateinamen in ein bestimmtes Muster passen, verwenden Sie niemals Muster in Zusammenhang mit Befehlen, die Dateien modifizieren oder zerstören (löschen) könnten!

Muster unterscheiden sich von den normalen Namen von Dateien und Dateiverzeichnissen durch besondere Zeichen (Buchstaben), die in ihnen auftreten. Diese Sonderzeichen sind *( ) ? % #* und der senkrechte Strich | . Es ist deshalb wenig ratsam, wenn auch nicht unmöglich, diese Zeichen in Dateinamen zu verwenden. Jedes dieser Sonderzeichen hat bei der Bildung von Mustern eine bestimmte Bedeutung. Einige sind Platzhalter für andere Buchstaben, andere fügen Teile von Mustern zu einem großen Muster zusammen. Muster werden gebildet, indem man »normale« Buchstaben und Sonderzeichen zu einem Muster zusammenfügt.

Der einfachste Platzhalter ist das Fragezeichen (?). Kommt in einem Muster ein Fragezeichen vor, so passen alle Dateinamen in dieses Muster, bei denen an der Stelle, wo im Muster das Fragezeichen steht, ein beliebiger Buchstabe vorkommt. In das Muster *H?ll?* passen zum Beispiel die Dateinamen *Hallo*, *Hello* und *Holla* (aber nicht *Heillo*, da hier zwei Buchstaben an der Stelle des Fragezeichens im Muster stehen). Und in das Muster A?B passen alle dreibuchstabigen Dateinamen, deren erster Buchstabe ein A und deren letzter Buchstabe ein B ist. Die Groß- und Kleinschreibung der Buchstaben in einem Muster oder im Dateinamen spielt dabei, wie im CLI üblich, keine Rolle. Auch der Name *aXb* paßt in das Muster *A?B*.

Der Platzhalter % hingegen paßt nur auf den Leerraum zwischen zwei Buchstaben (nicht zu verwechseln mit der Leerstelle zwischen zwei Wörtern). Er wird Ihnen im Augenblick wahrscheinlich sinnlos erscheinen, erhält seinen Sinn aber zusammen mit anderen Sonderzeichen in einem Muster.

Die restlichen Sonderzeichen außer ? und % dienen zum Kombinieren von größeren Mustern aus anderen kürzeren Mustern. Hierzu brauchen Sie nur 4 Regeln. Diese Regeln sind recht formal und abstrakt, und wenn sie nicht gleich verständlich für Sie sind, machen Sie sich nichts draus! Am Ende dieses Abschnittes folgen eine ganze Reihe von Beispielen auch für komplexere Muster. Diese sollten völlig ausreichen, Ihnen einen Eindruck davon zu geben, wie man Muster aufbaut.

Sind <m1> und <m2> zwei Muster, dann ist <m1><m2> ein Muster, in das alle Dateinamen passen, deren erster Teil in das Muster <m1> paßt und deren »restlicher« Teil dann in das

Muster <m2> paßt. Aus den beiden Mustern A?B und X??Z kann man zum Beispiel das Muster *A?BX??Z* bilden, in das unter anderem die Dateinamen *AyBXikZ* und *AlBXpoZ* passen. (Ich gebe zu, daß ich mir schönere Beispielnamen hätte einfallen lassen können. Vielleicht hätte man dann aber die Bedeutung der Sonderzeichen nicht mehr so deutlich gesehen.)

Das Doppelkreuz # ist das Wiederholungszeichen. Wenn <m> ein Muster, dann ist #<m> ein Muster, in das alle Texte passen, die aus Null, einer oder mehr Wiederholungen von <m> bestehen. In das Muster *A#B* passen also zum Beispiel die folgenden Dateinamen, die mit einem *A* beginnen, dem eine beliebige Anzahl *B* folgt: *A*, *AB*, *ABB*, *ABB*, und so weiter.

Der senkrechte Strich | entspricht in etwa dem Wort *oder*. Sind <m1> und <m2> zwei Muster, dann ist <m1> | <m2> ein Muster, in das alle Dateinamen passen, die in das Muster <m1> oder in das Muster <m2> passen. In das Muster *Ja|Nein* passen also die beiden Dateinamen *Ja* und *Nein*.

Die runden Klammern ( ) dienen dazu, auch größere Muster zu bilden, die die einzelnen oben vorgestellten Möglichkeiten nutzen. Klammern fassen das Muster, das Sie enthalten, zu einer »Einheit« zusammen. So werden größere Muster einerseits lesbarer und andererseits auch eindeutig genug, damit der Amiga sie verstehen kann. Zum Beispiel sind *A#AB* und *A#(AB)* völlig unterschiedliche Muster. In das Muster *A#AB* passen die Dateinamen *AB*, *AAB*, *AAAB*, *AAAAAAAB* und so weiter. In das Muster *A#(AB)* hingegen passen die Namen *AAB* (der auch in das erste paßt), *AABAB*, *AABABABABAB* und so weiter; also alle Namen, die mit *A* beginnen und dann mit einer beliebig langen Folge der beiden Buchstaben *AB* enden. Die Klammern im Muster *A#(AB)* ändern also völlig dessen Bedeutung gegenüber demselben Muster ohne Klammern.

Im Zusammenhang mit Klammern und dem | wird auch das % sinnvoll. Das Muster *(XY|%)ABC* zum Beispiel trifft auf die beiden Dateinamen *XYABC* und *ABC* zu – genauso wie das gleichbedeutend aber längere Muster *XYABC|ABC*.

Das letzte Sonderzeichen, das Hochkomma ‘, dient schließlich dazu, die besondere Bedutung der Sonderzeichen aufzuheben. Schreiben Sie direkt vor einem Sonderzeichen ein Hochkomma, dann können Sie so auch nach Dateinamen suchen, die dieses Sonderzeichen enthalten. In das Muster A#(‘?) zum Beispiel passen die Dateinamen A, A?, A?????? und so weiter.

Im folgenden werden nun noch ein paar Beispiele für Muster angeführt, die Ihnen hoffentlich eine Vorstellung davon geben, was mit Mustern möglich ist. Achten Sie bitte besonders auf die Bedeutung von Klammern für den Aufbau von Mustern.

Muster	**passende Dateinamen**
A?B	AAB, AbB, AcD, A1B, A2B
A#BC	AC, ABC, AbBC, ABbBc
A#(BC)	A, ABC, ABCbc, AbCBcBC, Abcbcbcbc
A#(B|C)D	AD, ABD, ACD, ABBD, ACCD, ABCD, ACBD
A#?B	AB, A9B, ASchubiduB, AXYZ17QRXB
A(B|%)#C	A, AB, ABC, AC, ACC, ABCCCCC
#?.TEXT	Brief.TEXT, alpha.TEXT, X1.TEXT
brief#?	brief, brief2, briefAnMama, »Brief an Mama«
'?#?'#	?#, ?AB#, ?XYZ#

Tabelle 7.1: *Beispiel für die Verwendung von Mustern für Dateinamen*

Probieren Sie einfach selbst ein paar Muster aus. Machen Sie dazu zunächst das Directory c (mit den CLI-Kommandos) zum aktuellen Directory:

```
1>cd :c
```

Wenn Sie nun den folgenden Befehl eingeben, erhalten Sie eine Liste aller CLI-Kommandos, die mit einem *S* beginnen (es sind einige):

```
1>dir s#?
```

Der folgende Befehl bringt hingegen die Kommandos zum Vorschein, deren Name an zweiter Stelle ein *E* enthält:

```
1>dir ?e#?
```

Und das letzte dieser kleinen Beispiele schließlich zeigt Ihnen eine Liste aller Befehle, die mit *E* enden:

```
1>dir #?e
```

Wie Sie an allen diesen Beispielen gesehen haben, ist besonders die Kombination *#?* sehr praktisch. Diese sollten Sie sich deshalb unbedingt einprägen! Sie können Muster aber natürlich nicht nur beim *dir*-Befehl verwenden. Sie sind auch für das Löschen und Kopieren ganzer Gruppen von Dateien oft sehr wichtig. Erzeugen Sie nun bitte fünf Dateien mit den Namen *test1*, *test2*, *test3*, *test4* und *test5* aus der Datei *Disk*.Info.

```
1>cd :
1>copy Disk.Info test1
1>copy test1 test2
1>copy test1 test3
1>copy test1 test4
1>copy test1 test5
```

Sie können alle diese Dateien nun mit einem Schlag in ein anderes Directory kopieren. Geben Sie dazu den folgenden Befehl ein:

```
1>copy test#? Utilities
```

Der *copy*-Befehl listet dann die Dateinamen in der Reihenfolge auf, in der sie kopiert werden und meldet hinter jedem Namen *copied* (kopiert), wenn er mit der jeweiligen Datei fertig ist (die Reihenfolge, in der die Namen auftauchen, kann bei Ihnen allerdings auch eine andere sein).

```
A500 WB 1.2 D:test3..copied
A500 WB 1.2 D:test2..copied
A500 WB 1.2 D:test1..copied
A500 WB 1.2 D:test4..copied
A500 WB 1.2 D:test5..copied
```

Die soeben kopierten Dateien können Sie nun auch mit einem einzigen Befehl löschen:

```
1>delete test#?
```

Auch der *delete*-Befehl listet daraufhin die Dateinamen in der Reihenfolge auf, in der er die Dateien löscht und meldet zu jeder Datei *Deleted*, wenn er damit fertig ist.

```
A500 WB 1.2 D:test3  Deleted
A500 WB 1.2 D:test2  Deleted
A500 WB 1.2 D:test1  Deleted
A500 WB 1.2 D:test4  Deleted
A500 WB 1.2 D:test5  Deleted
```

Nun müssen Sie nur noch die Kopien dieser Testdateien löschen, die Sie ja mit *copy* in das Directory *Utilities* gelegt haben. Geben Sie dazu den folgenden Befehl ein:

```
1>delete #?/test#?
```

Dieser Befehl ist besonders interessant, weil er »Directory-übergreifend« wirkt. Er durchsucht alle Directories (mit dem Teil *#?/* des Musters) im aktuellen Directory (in diesem Falle dem Wurzeldirectory *:*) und löscht in diesen Directories alle Dateien, die mit *test* beginnen und mit einem beliebigen Text enden (mit dem Teil *test#?* des Musters).

Leider können Sie nicht bei allen CLI-Kommandos Muster verwenden. Das folgende Kapitel enthält eine Aufstellung aller CLI-Befehle. Bei allen Befehlen, die Muster »verstehen«, wird dies ausdrücklich hervorgehoben und auch beschrieben, bei welchen Parametern und wie Sie Muster einsetzen können.

7.6 Directories – zum zweiten

Nach diesem kleinen Exkurs in die wichtigsten CLI-Kommandos werden Sie nun noch einmal mit theoretischen Grundlagen gequält, bevor Sie weitere neue Kommandos kennenlernen. Es geht dabei zum zweiten Mal um Directories (Dateiverzeichnisse), woraus Sie ersehen können, daß dieses Thema sehr wichtig für den Umgang mit dem CLI ist. Nachdem Sie ja schon wissen, was ein aktuelles Directory ist, werden Sie unter anderem nun auch erfahren, wie Sie es festlegen beziehungsweise ändern können.

7.6.1 Wie Sie das aktuelle Dateiverzeichnis wechseln

Mit dem CLI-Befehl *cd* (für *change directory = setze Dateiverzeichnis*) kann ein beliebiges Dateiverzeichnis zum *aktuellen Dateiverzeichnis* erklärt werden. Geben Sie zunächst einmal den folgenden Befehl ein (nur die erste Zeile; die zweite Zeile zeigt Ihnen, was daraufhin auf Ihrem Bildschirm erscheinen sollte):

```
1>cd
CLI-Disk:
```

Falls die Diskette im internen Laufwerk einen anderen Namen als *CLI-Disk* hat, erscheint natürlich dieser Name. Wenn Sie nämlich *cd* ohne jeden Parameter verwenden, wird der Name des aktuellen Directory ausgegeben. Sie können so jederzeit feststellen, »wo Sie sich im Dateienbaum gerade befinden«. Wenn Sie hingegen das aktuelle Directory wechseln wollen, müssen Sie den Pfadnamen des Directory, das das neue aktuelle Directory werden soll, hinter cd (als Parameter) angeben. Probieren Sie einmal den folgenden Befehl aus:

```
1>cd df0:c
```

Danach ist das Directory *c* auf der Diskette im internen Laufwerk (df0) das aktuelle Directory. Dieses Directory enthält die CLI-Kommandos, wovon Sie sich sofort überzeugen können, indem Sie einfach *dir* (und dann [Return]) eingeben. Auch wenn Sie jetzt *cd* ohne Parameter eingeben, meldet das CLI Ihnen, daß das aktuelle Directory nun *c* heißt. Sie können sofort wieder zum Wurzeldirectory zurückkehren, indem Sie die folgende Zeile eintippen:

```
1>cd df0:
```

Je nachdem, ob und wie Sie den *cd*-Befehl einsetzen, gibt es ganz verschiedene Wege »sich einer Datei« zu nähern. Sie können sich mit dem *cd*-Befehl Schritt für Schritt herantasten oder auch sofort den kompletten Namen verwenden. Die folgenden vier Befehlsfolgen zum Beispiel leisten auf verschiedene Art alle dasselbe. (Überlegen Sie sich einmal in Ruhe, warum.)

```
1>type df0:texte/privat/Brief1
1>cd df0:texte/privat
1>type Brief1
```

```
1>cd df0:
1>cd texte/privat
1>type Brief1
```

```
1>cd df0:texte
1>cd privat
1>type Brief1
```

7.6.2 Auf und ab im Dateienbaum

Sie haben bislang zwei Alternativen kennengelernt, eine Datei auf einer beliebigen Diskette eindeutig zu beschreiben. Die eine Alternative ist der vollständige Pfadname (zum Beispiel *archiv:texte/privat/Meier*), wobei Sie sich den Diskettennamen schenken können, wenn sich die Datei auf der aktuellen Diskette befindet (zum Beispiel *:texte/privat/Meier*). Die zweite Alternative ist der Dateiname allein (zum Beispiel *Meier*), der immer dann verwendet werden kann, wenn sich diese Datei im aktuellen Directory befindet. Hierzu müssen Sie gegebenenfalls aber erst einmal das aktuelle Directory ändern, um eine bestimmte Datei mit ihrem kurzen Namen bezeichnen zu können. Es gibt jedoch noch eine dritte Möglichkeit, Dateien anzusprechen, die sich nicht im aktuellen Directory befinden. Diese Möglichkeit ergibt ebenfalls einen Pfadnamen (jedoch nicht den »vollständigen Pfadnamen«). Dieser zweite Pfadname einer Datei ist oft kürzer als der vollständige. Er ist allerdings manchmal komplizierter und nicht so leicht zu bilden.

Zunächst einmal gilt, daß der Pfadname für die Dateien abgekürzt werden kann, die sich im aktuellen Directory oder in einem Directory befinden, das im aktuellen Directory (und noch weiter unten im Dateienbaum) liegt. Vor jeden Dateinamen setzt das CLI nämlich immer den Namen des aktuellen Dateiverzeichnisses, um den vollständigen Pfadnamen der Datei zu erhalten, sofern der eingegebene Name nicht mit einem Diskettennamen oder Doppelpunkt beginnt.

Das bedeutet zum Beispiel, daß die Datei *archiv:texte/privat/Meier* auch *privat/Meier* genannt werden kann, wenn das aktuelle Directory *archiv:texte* ist (*archiv:texte +privat/Meier = archiv:texte/privat/Meier*). Auf diese Weise können Sie zwar alle Dateien und Dateiverzeichnisse benennen, die hierarchisch »tiefer« liegen, aber nicht die, die hierarchisch höher oder in anderen Verzweigungen des Dateienbaums liegen.

In sehr fein und tief verästelten Dateibäumen kann es aber unpraktisch und schwierig sein, solche Dateien mit ihrem kompletten Pfadnamen angeben zu müssen. Wenn der Weg von der Wurzel zur gesuchten Datei sehr weit ist, kann es weniger Tipparbeit bedeuten, den Weg zu einer bestimmten Datei, ausgehend vom aktuellen Dateiverzeichnis, zu beschreiben. Auch dies ist möglich. Ähnlich wie das Zeichen Doppelpunkt (:) zur Wurzel der Dateiverzeichnisse auf einer Diskette führt und man einen Namen dann ausgehend von dieser Wurzel angeben kann, gibt es ein Zeichen, mit dem man im Baum nur »eine Stufe höher«, also in das über dem aktuellen Directory liegende Directory kommt. Schreibt man zu Beginn eines Dateinamens einen Schrägstrich, so entspricht dieser Schrägstrich dem Dateiverzeichnis, in dem das

aktuelle Dateiverzeichnis liegt. Dies kann man verwenden, um »im Baum höherzusteigen« oder auch, um zunächst höherzusteigen und dann wieder in einem Nachbar-Ast »abzusteigen«.

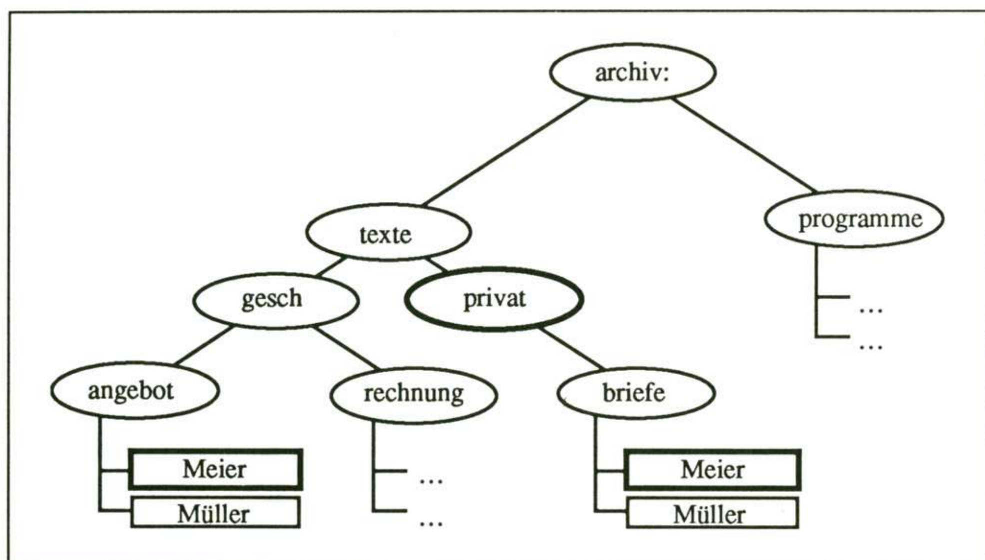


Bild 7.7: *Auf und Ab im Dateienbaum*

Betrachten Sie dazu noch einmal das vorangehende Bild 7.7. Wenn das aktuelle Directory *privat* heißt, kann die Datei *Meier* im Directory *angebot* ja ohne Ausnutzung des Schrägstrichs nur erreicht werden, indem man mit *texte/gesch/angebot/Meier* im Baum ganz nach oben geht und von der Wurzel aus den Namen bildet. Mit der besonderen Funktion des Schrägstrichs könnten Sie aber auch schreiben */gesch/angebot/Meier*. Der Schrägstrich führt zunächst in das Directory *texte* über dem aktuellen Directory. Von diesem aus geht es dann ins Directory *gesch*, dann nach *angebot* und in diesem schließlich findet sich die Datei *Meier*. Der Schrägstrich kann besonders dann, wenn sich zwischen dem aktuellen Directory und der Wurzel schon sehr viele Directory befinden, sehr viel Tipparbeit sparen.

Indem Sie an den Beginn eines Dateinamens mehrere Schrägstriche schreiben, können Sie übrigens auch mehrere Stufen im Baum »nach oben klettern«. So könnten Sie vom aktuellen Directory *privat* aus, die Datei *Meier* im Directory *angebot* auch mit *//texte/gesch/angebot/Meier* bezeichnen, was allerdings gegenüber *:texte/gesch/angebot/Meier* keinen Vorteil darstellt – es illustriert nur eine mögliche Verwendung des Schrägstrichs am Anfang eines Dateinamens.

7.6.3 Wie Sie neue Directories erzeugen und löschen

Wie Sie sich durch den Dateienbaum durchhangeln, sollten Sie nun wissen – zumindest so ungefähr. Jetzt brauchen Sie nur noch etwas Übung und Sie verstehen auch die längsten Pfadnamen. In diesem Abschnitt werden Sie noch erfahren, wie Sie neue Directories erzeugen und überflüssig gewordene wieder löschen können. Dann können Sie im CLI schon alle Arbeiten durchführen, für die Sie bislang die Workbench benötigten.

Auf der Workbench konnten Sie neue Schubladen nur dadurch erzeugen, indem Sie vorhandene duplizierten. Speziell zu diesem Zweck befindet sich ja die Schublade *Empty* auf der Workbench-Diskette. Im CLI sieht das etwas anders aus. Wie Sie vielleicht schon erwartet haben, gibt es auch für das Erzeugen von neuen Directories im CLI einen speziellen Befehl. Er heißt *makedir* (engl. *make directory = mache/erzeuge ein Dateiverzeichnis*). Sein einziger Parameter ist der Pfadname des neuen Directory, das Sie einrichten wollen. Wenn Sie den folgenden Befehl eingeben, erzeugen Sie zum Beispiel ein Directory namens *neudir* im Wurzeldirectory (:) der aktuellen Diskette. Probieren Sie es aus:

```
1>makedir :neudir
```

Und die folgenden vier Befehle legen zwei neue Directories im Directory *neudir* an und legen eine Kopie der schon mehrfach für diesen Zweck »mißbrauchten« Datei *Disk.info* unter dem Namen *Test* in das neue Directory *neu3*:

```
1>makedir :neudir/neu2
1>cd :neudir
1>makedir neu3
1>copy :Disk.Info neu3/test
```

Zum Löschen von Directories können Sie den Ihnen schon bekannten Befehl *delete* verwenden. Mit *delete* können Sie sowohl Dateien als auch Directories löschen – wenn letztere leer sind. Davon können Sie sich sofort überzeugen. Versuchen Sie dazu einmal, das Directory *neu3*, das Sie eben erzeugt haben, wieder zu löschen:

```
1>delete neu3
```

Sie erhalten daraufhin eine Fehlermeldung, die nichts anderes besagt, als daß Sie nur leere Directories löschen können. Mit den folgenden beiden Befehlen hingegen können Sie *neu3* wirklich löschen.

```
1>delete neu3/test
1>delete neu3
```

Wenn ein Directory sehr viele Dateien enthält, kann es aber recht lästig werden, diese alle nacheinander zu löschen, bevor man das Directory und seinen Inhalt wirklich los ist. Da scheint der *Discard*-Befehl der Workbench (oder der Mülleimer) schon wesentlich praktischer zu sein. Aber natürlich kennt auch das CLI – oder besser der *delete*-Befehl – eine Abkürzung, mit der Sie auf einen Schlag ein ganzes Directory löschen können. Die folgenden beiden Befehle

löschen alle Dateien und Directories, die Sie in diesem Abschnitt erzeugt haben (wovon Sie sich mit dem *dir*-Befehl überzeugen können):

```
1>cd :
1>delete neudir ALL
```

Wenn Sie nämlich am Ende eines *delete*-Befehls das Schlüsselwort ALL verwenden und der einzige Parameter von *delete* der Name eines Directory ist, werden alle Directories, die darin liegen und alle Dateien in allen Directories gelöscht. Der *delete*-Befehl listet während dieses Vorgangs die Namen der gerade gelöschten Dateien auf. Wenn Sie bemerken, daß darunter Namen von Dateien auftauchen, die Sie gar nicht löschen wollten, können Sie mit [Ctrl]-[C] versuchen, »zu retten, was zu retten ist«. Der *delete*-Befehl wird dann sofort abgebrochen. Die bis dahin schon gelöschten Dateien sind allerdings verloren.

7.6.4 Geräte – nicht nur Wurzeln der Dateienbäume

Weiter oben haben Sie ja bereits erfahren, daß Sie die Wurzel des Dateiverzeichnisses einer Diskette entweder mit dem Namen der Diskette – wie er auf der Workbench zu sehen ist – bezeichnen können, oder mit dem Namen des Diskettenlaufwerks, in dem sich die Diskette befindet. Sie können einen Pfadnamen zum Beispiel mit *DF0:* beginnen, wenn sich die Diskette im internen und mit *DF1:*, wenn sie sich im ersten externen Laufwerk befindet. *DF0:* und *DF1:* (oder besser *DF0* und *DF1*) sind »Gerätenamen«. Nicht nur Diskettenlaufwerke, sondern auch der Drucker, der Bildschirm, die Tastatur und einige andere Geräte haben solche Namen. Alle diese Geräte muß man vom CLI ja mit einem Namen ansprechen können, wenn man sie für einen Befehl benötigt. (In der Workbench haben Sie es oft leichter – auf eine Diskette kann man zum Beispiel einfach zeigen.)

Amiga-DOS und das CLI behandeln solche Geräte – wenn es sich nicht gerade um Diskettenlaufwerke handelt – wie spezielle Dateien. Sie können deshalb bei vielen CLI-Befehlen statt einem Dateinamen auch einen Gerätenamen als Parameter verwenden. Das CLI »erkennt« Geräte daran, daß hinter ihrem Namen immer ein Doppelpunkt steht. So ist es zum Beispiel möglich, eine Datei auf das Drucker-Gerät (also den Drucker) zu kopieren, wodurch sie natürlich ausgedruckt wird. Dies geht genauso wie das Kopieren einer Datei. Dazu müssen Sie nur statt des zweiten Dateinamens im Kopierbefehl den Gerätenamen *PRT:* (engl. für *printer = Drucker*) angeben. Der Befehl in der folgenden Zeile druckt zum Beispiel die Datei *startup-sequence* im Directory *s* der Workbench-Diskette aus. (Wenn Sie keinen Drucker angeschlossen und eingeschaltet haben, sollten Sie diesen Befehl nicht ausprobieren, weil sonst nur ein Requester mit einer Fehlermeldung erscheint, die nichts anderes besagt, als daß der Amiga keinen Drucker finden kann.)

```
1>copy :s/startup-sequence to PRT:
```

Geräte sind aber wirkliche »spezielle« Dateien, weil man sie meist nicht löschen kann (wie soll man einen Drucker löschen?), sie manchmal nur schreiben und nicht wieder lesen kann (wie soll man Daten von einem Drucker lesen?) und eine Reihe anderer Operationen, die mit Dateien möglich sind, nicht oder nur eingeschränkt verwendet werden können.

7.7 Ein-/Ausgabe-Umleitung

Die Gerätenamen, die Sie im letzten Abschnitt kennengelernt haben, spielen eine bedeutende Rolle im Zusammenhang mit einem weiteren sehr wichtigen Konzept des CLI: der sogenannten »Ein-/Ausgabe-Umleitung« (engl. *I/O redirection*). Viele Befehle des CLI lesen ihre Eingaben – sofern sie überhaupt welche benötigen – von einer Datei und schreiben alle Ausgaben in eine andere. Oft werden die Eingaben dabei von der Tastatur erwartet und die Ausgaben erscheinen im CLI-Fenster. (Auch die Tastatur und das CLI-Fenster sind eine Datei – oder genauer: ein Gerät namens *CON:*.)

Den Programmen ist es jedoch in der Praxis meist gleichgültig, von welchen Dateien beziehungsweise Geräten sie ihre Eingaben erhalten und wo sie ihre Ausgaben hinschicken. Deshalb kann man bei den meisten Programmen diese Dateizuordnungen von Ein- und Ausgabe gefahrlos »umlegen«, zum Beispiel auf den Drucker oder in eine Diskettendatei. Sie können auf diese Weise zum Beispiel den Inhalt eines Dateiverzeichnisses mit *dir* auf den Drucker schicken oder in einer Datei abspeichern, die Sie nachträglich mit einem Textverarbeitungsprogramm bearbeiten können.

7.7.1 Wie Sie Ein- und Ausgaben eines CLI-Kommandos umleiten

Um einem CLI-Kommando eine andere Eingabedatei zuzuweisen als die Tastatur, müssen Sie sofort hinter dem Befehlsnamen ein Kleinerzeichen (<) und danach den Namen der Datei oder des Gerätes schreiben, von dem die Eingaben kommen sollen. Genauso können Sie alle Ausgaben eines Befehls auf eine Datei oder ein Gerät schicken, indem Sie sofort hinter den Namen des Befehls ein Größerzeichen (>) und dann den Namen der Datei beziehungsweise des Geräts schreiben, in die (das) die Ausgaben gehen sollen. Wenn der Befehl diese Schlüsselwörter kennt, können Sie statt < auch FROM und statt > TO verwenden. Beim Befehl *copy* können Sie zum Beispiel beide Möglichkeiten, sowohl < als auch FROM verwenden.

Hierzu nun ein paar kleine Beispiele: Wenn Sie den folgenden Befehl eingeben, wird das Directorylisting nicht auf dem Bildschirm erscheinen, sondern geht in eine Datei namens *dirliste*.

```
1>dir >dirliste
```

Das Diskettenlaufwerk arbeitet dann eine ganze Weile und doch erscheint kein Dateiverzeichnis auf dem Bildschirm. Sie können es sich aber anschauen, indem Sie nun die Datei *dirliste* mit Hilfe von *type* ausgeben.

```
1>type dirliste
```

Alternativ dazu können Sie *dirliste* aber auch im *Notepad* einlesen und dort bearbeiten. Wie Sie diese Ausgaben eines Befehls umlenken, dürften Sie nun also wissen. Ein Beispiel für die gleichzeitige Umleitung von Ein- und Ausgabe sollen Sie aber nun auch noch kennenlernen: Nehmen wir einmal an, es gäbe einen Befehl namens *date* (es gibt ihn tatsächlich), der seine Eingabe von der Tastatur erwartet und seine Ausgaben in das CLI-Fenster schickt. Tippen Sie

aber die folgende Befehlszeile, so holt er seine Eingaben von der Datei *Input* und druckt seine Ausgaben auf dem Drucker aus (sofern Sie einen angeschlossen und eingeschaltet haben).

```
1>date <Input >PRT:
```

Den vollen Nutzen der Ein-/Ausgabe-Umleitung können Sie wahrscheinlich aber erst abschätzen, wenn Sie wirklich täglich mit dem CLI umgehen. Solche Umleitungen lassen sich sehr vielseitig einsetzen und sind deshalb eine Möglichkeit, die seit einigen Jahren in keinem neuen Betriebssystem für Computer mehr fehlen darf.

7.7.2 Ein-/Ausgabe-Umleitung auf Geräte und Dateien

Sie sollten bei der Ein-/Ausgabe-Umleitung aber den grundsätzlichen Unterschied zwischen einer Umleitung auf eine Datei und ein Gerät beachten. Beides dient meist völlig unterschiedlichen Zwecken. So wird zum Beispiel die Ausgabe-Umleitung auf eine Datei meist deshalb vorgenommen, um eine »flüchtige« Ausgabe, die schnell über den Bildschirm huschen würde, permanent abzuspeichern, um sie in Ruhe betrachten oder mit anderen Programmen weiterverarbeiten zu können. Eine Umleitung auf ein Gerät hingegen erweitert fast immer die Funktionalität eines Befehls. Sie können zum Beispiel ein Directory nicht nur anzeigen, sondern auch ausdrucken lassen, indem Sie es mit dem folgenden Befehl zum Drucker schicken:

```
1>dir >PRT:
```

Übrigens hätten sich die Entwickler des CLI wegen der Ein-/Ausgabe-Umleitung eigentlich auch den *type*-Befehl sparen können. Der *type*-Befehl macht ja nichts weiter, als den Inhalt einer Textdatei auf den Bildschirm zu kopieren (!) – das kann der *copy*-Befehl natürlich auch. Die beiden folgenden Befehlsfolgen sind deshalb »äquivalent«, das heißt, sie leisten dasselbe:

```
1>dir >dirliste
1>type dirliste
1>type >PRT: dirliste
1>dir >dirliste
1>copy >* dirliste
1>copy dirliste TO PRT:
```

Der erste Befehl schreibt jeweils das Dateiverzeichnis des aktuellen Directory in die Datei *dirliste*. Der zweite Befehl listet diese Datei dann auf dem Bildschirm und der dritte auf dem Drucker (diesen letzten Befehl sollten Sie fortfallen lassen, wenn Sie keinen Drucker angeschlossen und eingeschaltet haben). Sie wundern sich vielleicht über den Befehl *copy >* dirliste*. In diesem wird die Ausgabe des *copy*-Befehls auf das Gerät * (Sternchen) umgelenkt. Das ist ein ganz besonderes Gerät: das CLI-Fenster, in dem Sie gerade arbeiten. Deshalb erscheint die Datei *dirliste* (das Dateiverzeichnis) daraufhin am Bildschirm.

7.7.3 Eine Liste der »echten« Geräte

Einige Geräte und Gerätenamen, die Sie im CLI verwenden können, haben Sie nun schon kennengelernt. Amiga-DOS und damit das CLI kennt aber insgesamt 14 »echte« Geräte (die »unechten« werden Sie weiter unten noch kennenlernen). Die folgende Tabelle enthält eine Aufstellung ihrer Namen und jeweils ein Stichwort zu ihrer Bedeutung.

Gerätename	Bedeutung
DF0:	Internes Diskettenlaufwerk
DF1:	1. externes Diskettenlaufwerk
DF2:	2. externes Diskettenlaufwerk
DF3:	3. externes Diskettenlaufwerk
DH0: oder HD0:	1. Festplattenlaufwerk
DH1: oder HD1:	2. Festplattenlaufwerk
RAM:	Pseudo-Diskettenlaufwerk im RAM-Speicher
NIL:	Leeres Pseudogerät
SER:	Serielle Schnittstelle
PAR:	Parallele Schnittstelle
PRT:	Drucker
CON:	Konsole (Bildschirm und Tastatur)
RAW:	Spezielle Konsole
*	CLI-Fenster

Tabelle 7.2: Physikalische Geräte des Amiga

Wofür Sie die Namen der Diskettenlaufwerke benötigen, haben Sie bereits erfahren: Der vollständige Pfadname einer Datei beginnt immer mit dem Namen der Diskette/Festplatte oder mit dem Namen des Disketten-/Festplattenlaufwerks. Im Betriebssystem des Amiga sind insgesamt vier Diskettenlaufwerke (*DF0:* bis *DF3:*) und zwei Festplattenlaufwerke vorgesehen (*DH0:* und *DH1:* oder *HD0:* und *HD1:*). (Sie können diese natürlich nur dann in einem Pfadnamen verwenden, wenn sie auch angeschlossen sind.)

Ein ganz besonderes Diskettenlaufwerk ist auch noch *RAM:*. Das Gerät *RAM:* simuliert ein Diskettenlaufwerk beziehungsweise eine Diskette im RAM-Speicher des Amiga. Mehr über diese spezielle Diskette erfahren Sie im Abschnitt *Ein ganz spezielles Gerät: die RAM-Disk* dieses Kapitels.

NIL: ist ebenfalls ein ganz besonderes »Gerät«. Manchmal kann es nämlich auch sinnvoll sein, zum Beispiel für Testzwecke Daten »ins Nichts« zu schicken. Dazu dient *NIL:*. Kopiert man zum Beispiel eine Datei zum Gerät *NIL:*, so verschwinden die Daten auf Nimmerwiedersehen

in diesem Gerät. *NIL:* wird wahrscheinlich nur von fortgeschrittenen Benutzern und auch von diesen nur sehr selten benötigt. Auch der »Noch-nicht-Profi« kann *NIL:* aber sinnvoll einsetzen. Sie können mit dem folgenden Befehl nämlich erreichen, daß eine Datei komplett gelesen wird. Sie können so feststellen, ob die Datei physikalisch noch in Ordnung ist oder ob sie einen Diskettenfehler enthält, der ein Lesen verhindert (in diesem Fall erscheint eine entsprechende Fehlermeldung am Bildschirm).

```
1>copy dateiX to nil:
```

Etwas »bodenständiger« sind da schon die Geräte *SER:*, *PAR:* und *PRT:*. *SER:* bezeichnet die serielle Schnittstelle (Buchse) an der Rückseite des Amiga, *PAR:* die parallele Schnittstelle. Sie können verschiedene Geräte an diese Stecker anschließen, die sowohl Daten empfangen als auch senden können. Dies können zum Beispiel Drucker und Geräte zur Telekommunikation (Modems) sein. Die Software des Amiga sorgt dafür, daß sich solche Geräte wie Dateien verhalten, um etwas einfacher damit umgehen zu können. Je nachdem, welches Gerät Sie an welcher Schnittstelle angeschlossen haben, können Sie sowohl Ein- als auch Ausgaben nach *SER:* und *PAR:* umleiten. Von Ausnahmen abgesehen, werden Sie diese beiden Gerätenamen aber trotzdem sehr selten benötigen, da mit den Geräten, die Sie dort anschließen, meist auch Programme kommen, die sich um solche Details selbst kümmern. Ein Modem an der seriellen Schnittstelle werden Sie zum Beispiel nie benutzen, indem Sie eine Datei nach *SER:* kopieren. Sie werden statt dessen ein sogenanntes Terminal- oder DFÜ-Programm verwenden, für dessen Benutzung Sie gar nicht wissen müssen, daß das Modem am Gerät *SER:* hängt.

Das Gerät *PRT:* hingegen werden Sie vielleicht häufiger verwenden. *PRT:* steht für *printer* und bezeichnet immer den Drucker, unabhängig davon, an welche Schnittstelle Sie ihn angeschlossen haben. Kopieren Sie eine (Text-) Datei nach *PRT:*, so wird diese im allgemeinen ausgedruckt. *PRT:* kümmert sich dabei automatisch um solche Feinheiten wie den Typ des angeschlossenen Druckers, wie schnell man mit ihm kommunizieren kann, ob er farbig drucken kann, und so weiter. Verwenden Sie stets *PRT:*, wenn Sie einen Text ausdrucken wollen und nicht *SER:* oder *PAR:*, selbst wenn Sie genau wissen, daß Ihr Drucker am *SER:* angeschlossen wurde! Ansonsten müssen Sie natürlich auch noch beachten, daß *PRT:* nur Daten empfangen und keine abgeben kann – im Gegensatz zu SER: und PAR:, die, wenn die entsprechenden Geräte angeschlossen sind (keine Drucker), als Eingabe»dateien« verwendet werden können.

Wenn Sie vom CLI aus eine Datei nach *PRT:* schicken, sollten Sie aber darauf achten, daß es wirklich eine sogenannte »reine Textdatei« ist, die nur druckbare Buchstaben und Zeilenvorschübe enthält. Die Text-Editoren *ed* und *microEmacs*, die in einem folgenden Kapitel beschrieben werden, erzeugen zum Beispiel solche Dateien. Erstellen Sie mit einem speziellen Textverarbeitungsprogramm einen Text, in dem Sie verschiedene Farben und/oder Zeichensätze und Größen verwenden, so müssen Sie diesen im allgemeinen auch innerhalb des Textverarbeitungsprogramms ausdrucken. *PRT:* versteht solche Dateien nicht und liefert deshalb völligen Unsinn auf dem Papier.

Die Geräte *CON:* und *RAW:* entsprechen den »Geräten« des Amiga, mit denen Sie wohl am meisten zusammenarbeiten: der Tastatur und dem Bildschirm. Zusammengefaßt nennt man Tastatur und Bildschirm in der Computerbranche meist *Konsole* (engl. ***console***). *CON:* und *RAW:* können beide sowohl als Ein- als auch Ausgabedatei verwendet werden – sie vereinen Tastatur und Bildschirm in einem Gerät. Sie werden *CON:* und *RAW:* aber trotzdem recht selten verwenden. Sie können eine Textdatei zum Beispiel nicht auf den Bildschirm schicken, indem Sie sie nach *CON:* oder *RAW:* kopieren. Dafür ist das besondere Gerät * (Sternchen) gedacht, das das aktuelle CLI-Fenster bezeichnet. Diese Verwendung von * ist ja bereits im letzten Beispiel des vorangehenden Abschnitts demonstriert worden.

CON: und RAW werden hauptsächlich beim Erzeugen zusätzlicher CLI-Fenster benötigt. Ja, Sie haben richtig gelesen: »zusätzlicher CLI-Fenster«. Sie können mit dem dafür vorgesehenen Befehl *newcli* ein oder mehrere zusätzliche CLI-Fenster öffnen und zwischen diesen rasch hin und her wechseln. Mehr dazu aber im Abschnitt *Wie Sie parallel mit mehreren CLIs arbeiten*, weiter unten in diesem Kapitel.

7.8 Alternative Wege zu Programmen und Dateien

Wie Sie in den vorangegangenen Abschnitten erfahren haben, ist es nicht immer ganz einfach, im CLI eine Datei oder ein Directory zu bezeichnen. Während Sie auf der Workbench einfach nur mit der Maus ein Piktogramm anklicken, müssen Sie oft lange und komplizierte Pfadnamen bilden – oder sich eine Menge Tricks merken, um diese Namen abkürzen zu können. Im CLI gibt es deshalb auch noch zwei andere Methoden, mit denen nach Dateien gesucht wird, deren Namen Sie in seiner kurzen Form und nicht als Pfadnamen eingeben: »logische Geräte« und den »Suchpfad« für Programme.

7.8.1 »Logische« Geräte

Wenn Sie einen Befehl im CLI eingeben, so verwendet das CLI das erste Wort dieses Befehls als den Namen eines Programms. Um den Befehl auszuführen, wird zunächst dieses Programm gestartet – falls es gefunden wird. Unabhägig davon, wie gerade das aktuelle Directory und die aktuelle Diskette heißt, »findet« das CLI Programme, die im aktuellen Directory oder im Directory *c* der Workbench-Diskette liegen. Genauer gesagt, sucht das CLI im aktuellen Directory und im Wurzeldirectory des »logischen Laufwerks« *C:* nach Programmen. Amiga-DOS – und damit das CLI – kennt neben den physikalischen Geräten wie *DF0:* und *PRT:* nämlich auch noch logische Geräte oder noch besser gesagt: »logische Disketten- laufwerke«. Zu einem solchen logischen Diskettenlaufwerk gehört – genauso wie zu einem echten – ein Wurzeldirectory. Dieses Wurzeldirectory kann in Wirklichkeit ein beliebiges Directory auf einer beliebigen Diskette oder Festplatte sein.

Das CLI speichert dazu intern die Assoziation ab, daß als Wurzeldirectory des Laufwerks *C:* das Directory *<Startdiskette>:C* zu verwenden ist. Pfadnamen, die mit *C:* beginnen, werden dazu in Pfadnamen »übersetzt«, die mit *<Startdiskette>:C/* beginnen. Diese *<Startdiskette>* (die Workbench-Diskette, mit der der Amiga gestartet wurde), entspricht dem logischen Laufwerk *SYS:*. Statt den eventuell langen Namen der Startdiskette oder den Laufwerksnamen

zu verwenden, kann man bei der Angabe der Pfadnamen von Dateien, die auf der Startdiskette liegen, auch *SYS:* sagen.

Dies spart manchmal Tipparbeit. (Die Namen einiger anderer logischer Laufwerke sind bedeutend kürzer als die entsprechenden Pfadnamen zu den Directories.) Ein noch bedeutenderer Vorteil ist aber, daß die Zuordnungen von logischem Laufwerk zu wirklichem Dateiverzeichnis geändert werden können. Wenn Sie zum Beispiel die Startdiskette (Workbench) aus dem Diskettenlaufwerk herausnehmen und einen CLI-Befehl eintippen, dann schaut das CLI auf dem logischen Laufwerk *C:* nach dem entsprechenden Programm. Dazu muß aber die Diskette wieder eingelegt und eine andere entfernt werden. Das kann lästig sein! Befinden sich aber dieselben CLI-Befehle auf einer anderen Diskette, die sowieso gerade in einem Laufwerk ist, so können wir (mit einem entsprechenden Befehl) sagen: »Bitte, liebes CLI, das logische Laufwerk *C:* ist jetzt im Dateiverzeichnis *DF1:programme* zu finden!«. Besitzen wir gar eine Festplatte, so kann man mit einer ähnlichen Zuordnung erreichen, daß diese als Startdiskette aufgefaßt wird – und damit viele Vorgänge schneller ablaufen.

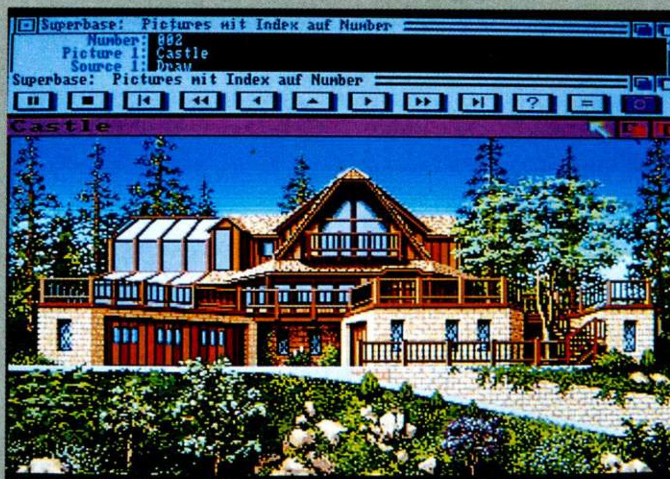
Welchem Dateiverzeichnis ein logisches Laufwerk zugeordnet ist, kann man zudem jederzeit mit Hilfe des CLI-Befehls *assign* ändern. Dies geht bei physikalischen Geräten natürlich nicht; *DF0:* ist immer das interne Laufwerk. Die Hauptfunktion logischer Laufwerke ist es, dem Betriebssystem des Amiga zu sagen, wo es bestimmte Dateien findet, die für den Betrieb benötigt werden. Hätte man festgelegt, daß diese Dateien immer in bestimmten Dateiverzeichnissen auf bestimmten Disketten zu finden sein müssen, müßte man Einbußen an Flexibilität hinnehmen. Beim Neustart des Amiga 500 werden nun zwar die logischen Laufwerke automatisch den Directories auf der Start-Diskette zugeordnet, in denen die jeweiligen Dateien zu finden sind. Danach kann man diese Zuordnungen aber jederzeit ändern.

Ein weiteres Beispiel für dieses Verhalten ist das Programm *Notepad*. Die Zeichensätze, die im *Notepad* zur Verfügung stehen, werden auf dem logischen Laufwerk *FONTS:* gesucht. Normalerweise ist dieses logische Laufwerk dem Dateiverzeichnis *fonts* auf der Startdiskette zugeordnet. Hat man aber zum Beispiel eine spezielle Diskette *FontDisk*, die andere Zeichensätze in einem Directory namens *Spezial* enthält, so kann man vor dem Aufruf von *Notepad* einfach den folgenden Befehl im CLI eingeben:

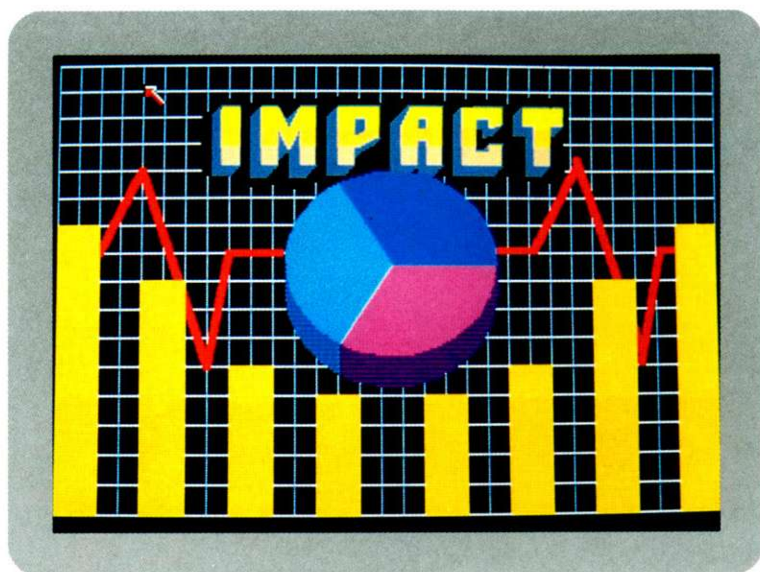
```
1>assign FONTS: FontDisk:Spezial
```

In Folge dieses Befehls wird *Notepad* die Zeichensätze, die es im Menü *Fonts* anbietet, dann auf der Diskette *FontDisk* im Dateiverzeichnis *Spezial* suchen. Auf ähnliche Weise kann man auch andere Stellen, an denen Amiga-DOS nach bestimmten Dateien sucht, ändern.

Die folgende Tabelle gibt einen Überblick über die logischen Geräte, die standardmäßig vom Amiga-DOS benutzt werden, sowie darüber, welchen Dateiverzeichnissen sie beim Start zugeordnet werden und wozu sie dienen. *StartDisk* bezeichnet in dieser Tabelle die Workbench-Diskette, mit der der Amiga 500 gestartet wurde oder die nach dem letzten Neustart mit [Ctrl]+[*C*=]+[*A*] eingelegt wurde.


Superbase: Pictures mit Index auf Number
Number: 002
Picture 1: Castle
Source 1: Draw
Superbase: Pictures mit Index auf Number
Castle

Quit
CLImate 1.2
13 directories 13 files in 'df2:'
Disk 199592 free
File
Disk 2928 free
Trashcan
.info
CLI
CLI.info
CLImate
CLImate.info
DiamondRing
DiamondRing.info
dozer.hires
Eiffel Tower
df0:
df0:
df2:
RAM:
c
Demos
devs
Empty
Expansion
fonts
l
libs
s
System
df2:
Read
Parent
Destination
Read
All
Clear
Copy
Move
Delete
Pattern
Print
Format
Batch
Rename
Display
ShowPic
Makedir


IMPACT

Color Palette
R G B H S V
15
12
8
4
0
SPREAD
Ex COPY
RANGE: C1 C2 C3 C4
SPEED
CANCEL
UNDO
OK

National Forecast
25
15
47
H
75
85

Gerätename	Dateiverzeichnis	Lagerort für
SYS:	StartDisk:	Workbench-Diskette
C:	SYS:c	CLI-Kommandos
L:	SYS:l	System-Libraries
S:	SYS:s	Kommandofolgen
LIBS:	SYS:libs	Libraries
DEVS:	SYS:devs	Gerätetreiber
FONTS:	SYS:fonts	Zeichensätze

Tabelle 7.3: Logische Geräte des Amiga

Diese Liste oder auch eine noch längere können Sie sich – ohne die Erläuterungen in der dritten Spalte natürlich – anzeigen lassen, indem Sie im CLI *assign* ohne Parameter eingeben und [Return] drücken. Sie erfahren dann, welche logischen Laufwerke im Augenblick existieren und welchen Directories auf welchen Disketten sie zugeordnet sind.

Einige der Bezeichnungen in der dritten Spalte dürften Ihnen (noch) nicht viel sagen. Hierzu müßten Sie noch mehr über Konzepte wissen, die erst im letzten Teil dieses Buches erläutert werden. Diese logischen Laufwerke werden der Vollständigkeit halber aber hier trotzdem schon mit aufgeführt.

Eine zentrale Bedeutung bei fast allen logischen Geräten nimmt *SYS:* ein. Dies ist der logische Laufwerkname für die Diskette, mit der der Amiga gestartet wurde (die »Workbench-Diskette«). Benötigt man irgendwelche Dateien oder Dateiverzeichnisse auf der Startdiskette, so kann man immer den Namen *SYS:* für das Wurzel-Dateiverzeichnis verwenden, egal in welchem Laufwerk die Diskette liegt, oder ob sie überhaupt in einem Laufwerk ist.

C: ist das logische Laufwerk, in dem die Programme gesucht werden, die die CLI-Befehle (*commands*) ausführen. Angenommen, Sie haben auf einer anderen Diskette als der Startdiskette einen vollständigeren Satz von CLI-Befehlen, der sich dort ebenfalls im Dateiverzeichnis *c* befindet. Der folgende Befehl sorgt dann dafür, daß das CLI immer, wenn Sie einen Befehl eintippen, dort nach dem entsprechenden Programm sucht (und überhaupt nicht mehr auf der Startdiskette):

```
1>assign C: AndereDisk:c
```

Die Geräte *L:* und *LIBS:* enthalten Programmbibliotheken (engl. *Libraries*), die vom System beziehungsweise von anderen Programmen genutzt werden können. Diese Bibliotheken werden erst dann für Sie interessant, wenn Sie beginnen, den Amiga 500 selbst zu programmieren. Es ist selten nötig, deren Zuordnungen zu einem Dateiverzeichnis zu ändern.

Im logischen Laufwerk *DEVS:* (für *devices* oder zu deutsch *Geräte*) befinden sich die *Gerätetreiber*. Deren Namen enden alle mit ».device«, wovon Sie sich mit Hilfe des *dir*-Befehls jederzeit überzeugen können. Gerätetreiber enthalten, ähnlich wie Bibliotheken,

kleine Programmteile, die von anderen Programmen benötigt werden. In diesem Fall helfen diese Programmteile, die verschiedenen Geräte, die man an die Schnittstellen an der Rückseite Ihres Amiga 500 anschließen kann, zu bedienen. Dies können ja sehr verschiedene Geräte sein, die man beim Schreiben der Betriebssystem-Software nicht schon alle vorhersehen konnte. Deshalb ist es sinnvoller, deren Steuerung auf solche Geräte-Treiber zu verlagern, die man problemlos austauschen kann, wenn Sie einmal ein neues Gerät bekommen.

Im Laufwerk *DEVS:* befindet sich auch das Directory *printers*, das die verschiedenen Druckertreiber enthält, die Ihnen im Programm *Preferences* zur Auswahl angeboten werden.

Das logische Laufwerk *FONTS:* enthält die Dateien, die die verschiedenen Schriftarten beschreiben, die Sie zum Beispiel in *Notepad* verwenden können. Sie wurden oben bereits erläutert. Auf dem Laufwerk S: schließlich befinden sich die »Kommandofolgen« oder »Kommandosequenzen«. Kommandofolgen bieten die Möglichkeit, eine Folge von CLI-Befehlen zu einem neuen Befehl zusammenzufassen. Wenn Sie versuchen, eine solche Kommandofolge aufzurufen und das CLI findet sie nicht im aktuellen Directory, so sucht es im Laufwerk *S:* danach. Mehr zu Kommandofolgen erfahren Sie noch in einem der folgenden Kapitel.

Viele der üblichen Zuordnungen von logischen Laufwerken zu Dateiverzeichnissen werden Sie kaum jemals ändern müssen. Es kann jedoch in einem Fall nötig werden, sie alle zu ändern. Falls Sie sehr viel RAM-Speicher in Ihrem Amiga oder gar eine Festplatte besitzen, so kann es sehr sinnvoll sein, diese als »Startdiskette« zu verwenden (obwohl Sie natürlich nicht von dem entsprechenden Gerät starten können). Dies ist vor allem deshalb »sehr sinnvoll«, weil dann alle Vorgänge wesentlich schneller ablaufen. Festplatten, und erst recht das RAM-Laufwerk, sind Disketten von der Geschwindigkeit her weit überlegen. Die folgende Befehlsfolge würde das Gewünschte erreichen. In dieser Befehlsfolge wird davon ausgegangen, daß Sie eine Festplatte, besitzen, die den Gerätenamen DH0: trägt. (Falls Sie wirklich stolzer Besitzer einer Festplatte sind, sollten Sie sich diese Folge von Befehlen aus Gründen der Arbeitsersparnis natürlich in einer Kommandofolge abspeichern.)

```
assign   SYS:     DH0:
assign   C:       DH0:c
assign   L:       DH0:l
assign   S:       DH0:s
assign   LIBS:    DH0:libs
assign   DEVS:    DH0:devs
assign   FONTS:   DH0:fonts
```

Sie können danach die wirkliche Startdiskette aus dem Diskettenlaufwerk nehmen und werden sie nie wieder benötigen, solange der Amiga eingeschaltet bleibt.

7.8.2 Der Suchpfad für Programme

Während das Konzept der »logischen Laufwerke« für die Suche nach Zeichensätzen, Gerätetreibern und so weiter recht praktisch ist, besitzt es für die Suche nach Programmen

eigentlich zuwenig Flexibilität. Wenn Sie sich zum Beispiel einmal kurz die Organisation der Workbench-Diskette vor Augen halten, die Sie zu Ihrem Amiga 500 mitgeliefert bekommen, werden Sie feststellen, daß sich in mindestens vier Directories Programme finden: im Wurzeldirectory (unter anderem *Preferences*), im Directory *c* (die CLI-Kommandos), in Directory *Utilities* (unter anderem *Notepad*) und im Directory *System* (unter anderem *IconEd*). Nur eines dieser Directories kann das logische Laufwerk *C:* sein und nur eines kann das aktuelle Directory sein. Programme in den beiden übrigbleibenden Directories findet das CLI nicht, wenn Sie nicht mit ihrem vollen Pfadnamen genannt werden.

Um das Auffinden von Programmen etwas flexibler zu gestalten und Programme an möglichst vielen Stellen ablegen und sie trotzdem mit kurzen Namen aufrufen zu können, gibt es den aktuellen Suchpfad für Programme. Dieser besteht aus einer Liste von Directories, die das CLI nacheinander durchsucht, um ein Programm zu finden, dessen Namen Sie eingegeben haben. Erst wenn in keinem dieser Directories ein Programm des angegebenen Namens gefunden werden kann, gibt das CLI eine Fehlermeldung aus.

Diesen Suchpfad können Sie sich mit dem Befehl *path* (engl. path = Weg/Pfad) anzeigen lassen, ändern und löschen. Geben Sie einmal die folgende Zeile ein und betrachten Sie die daraufhin erscheinende Liste:

```
1>path
Current Directory
A500 WB 1.2 D:System
C:
```

Ohne Parameter aufgerufen, listet *path* nämlich die Directories im Suchpfad in der Reihenfolge auf, in der sie das CLI »abklappert«, um ein Programm zu finden. Die zweite Form des *path*-Befehls dient dazu, ein weiteres Directory in den Pfad einzuhängen. Geben Sie beispielsweise die folgende Zeile ein und lassen Sie sich dann noch einmal (mit *path* allein ohne Parameter) den Suchpfad zeigen:

```
1>path SYS:fonts ADD
```

Sie haben auf diese Weise das Directory *SYS:fonts* (also *fonts* auf der Workbench-Diskette) dem Pfad hinzugefügt. Dieses Directory wird nun nach allen anderen Directories aber noch vor C: durchsucht, um Programme zu finden. Das hat allerdings keine Wirkung, da dieses Directory keine Programme enthält. Sie können auch mehrere Directories auf einmal zum Pfad hinzufügen. Schreiben Sie hierzu einfach bis zu zehn Directory-Namen hinter *path* und vergessen Sie nicht, diese Liste mit dem Schlüsselwort ADD (engl. *to add = ergänzen/ hinzufügen*) abzuschließen.

Normalerweise wird man möglichst viele Directories in den Pfad hängen. Es kann jedoch auch nötig werden, wieder Directories aus dem Pfad herauszunehmen. Wenn der Suchpfad zum Beispiel Directories auf Disketten enthält, die sich in keinem Laufwerk befinden, so kann es passieren, daß diese Disketten jedesmal angefordert werden, wenn Sie einen Befehl eingeben. In so einem Fall ist es nötig, den Suchpfad »auf Null zu setzen«. Geben Sie dazu den folgenden Befehl ein:

```
1>path RESET
```

Mit dem Schlüsselwort RESET werden alle zusätzlich durchsuchten Directories aus dem Suchpfad genommen. Danach wird nur noch das aktuelle Directory und das logische Laufwerk *C:* durchsucht, um ein Programm zu finden, das einen im CLI eingegebenen Befehl ausführen kann.

7.9 Wie Sie parallel mit mehreren CLIs arbeiten

Vielleicht ist es Ihnen noch nicht aufgefallen, aber das CLI hat (scheinbar) einen großen Nachteil gegenüber der Workbench. Bislang haben Sie immer gesehen, wie man dem CLI einen Befehl gibt, dann wartet, bis dieser fertig ist und erst dann einen neuen eingeben kann. Die Multitasking-Fähigkeiten des Amiga-Betriebssystems werden dabei nicht ausgenutzt. Auf der Workbench hingegen können Sie ein Anwendungsprogramm starten, dann auf ein Disketten-Piktogramm klicken und es öffnen (was einem *dir*-Befehl entspricht) und mit einem Mausklick wieder im Programm zurück sein. Das ist ja schließlich der Vorteil des Multitasking, zu dem der Amiga 500 fähig ist. Deshalb gibt es natürlich auch im CLI die Möglichkeit, mehrere Programme parallel zu bearbeiten.

Im Gegensatz zur Workbench startet das CLI ein Programm aber normalerweise nicht als parallelen Task. Sie können nicht gleichzeitig mit einem solchen Befehl arbeiten und dem CLI neue Befehle erteilen, wie es auf der Workbench ja üblich ist. Das CLI läuft in Wirklichkeit zwar weiter, es wartet jedoch immer auf die Beendigung eines einmal gestarteten Programms, bevor es einen neuen Befehl akzeptiert. Das ist aber nicht ganz so schlimm, wie es sich anhört, da die CLI-Befehle fast alle sehr kleine Programme sind, die auch recht schnell mit ihrer Aufgabe fertig sind.

Sollten Sie es aber sehr eilig haben, können Sie auch schon einen neuen Befehl eingeben, bevor der vorhergehende ganz fertig ist. Während Sie diesen Befehl eintippen, macht der gerade arbeitende Befehl eine Pause und fährt erst dann fort, wenn Sie die [Return]-Taste drücken. Das CLI wartet dann, bis das gerade laufende Programm beendet ist und führt dann den neuen Befehl aus. Das ist natürlich kein Multitasking, da immer nur ein Befehl ausgeführt wird und der nächste so lange warten muß, bis der vorangehende fertig ist. Es gibt aber auch Möglichkeiten für echtes Multitasking im CLI.

7.9.1 Mehrere CLI-Fenster

Die erste dieser Möglichkeiten ist die Verwendung mehrerer CLIs. Es hält Sie ja nichts davon ab, nach dem Öffnen des ersten CLI-Fensters zurück in die Workbench zu gehen, um das CLI-Piktogramm ein zweites Mal zu öffnen. Das ergibt natürlich ein zweites CLI-Fenster, in dem Befehle parallel zu Befehlen im ersten CLI ablaufen können. Wenn im ersten CLI zum Beispiel ein »länglicheres« Programm läuft (zum Beispiel der Compiler für eine Programmiersprache), können Sie einfach mit der Maus auf das zweite CLI-Fenster klicken und dort andere Befehle eingeben.

Neue CLI-Fenster können aber auch von einem CLI aus eröffnet werden. Hierzu dient der Befehl *newcli* (neues CLI). Geben Sie diesen Befehl versuchsweise zweimal ein.

```
1>newcli
1>newcli
```

Daraufhin erscheinen zwei neue CLI-Fenster. Legen Sie diese am besten etwas beseite und verkleinern Sie sie, damit Sie die Inhalte aller drei CLI-Fenster überblicken können. Vielleicht ist Ihnen beim Erscheinen der beiden neuen CLI-Fenster aufgefallen, daß das Prompt in diesen Fenstern etwas anders aussieht. Statt *1>* steht dort nun *2>* oder *3>*. Normalerweise zeigt jedes CLI nämlich vor dem Größer-Zeichen seine »laufende Nummer« an. Diese Nummer wird aufsteigend von 1 bei der Entstehung eines neuen CLIs »vergeben« und ändert sich auch dann nicht, wenn andere CLIs mit *endcli* beendet werden. (Sie können jedes beliebige CLI-Fenster mit *endcli* schließen.) Man kann dadurch immer eindeutig feststellen, mit welchem CLI man es »zu tun hat«.

Wenn Sie *newcli* aufrufen, erscheint ein neues Fenster am Bildschirm. Sie können dabei auch gleich als Parameter angeben, wie groß dieses Fenster werden und wie es liegen soll. Dazu wird die Angabe des Gerätes, von dem aus das CLI seine Befehle empfängt und auf dem es seine Meldungen ausgeben kann, benötigt. Dies wird normalerweise das CON:-Gerät (Bildschirm und Tastatur) sein. Sie können so durch die Angabe der entsprechenden Teile des vollständigen CON:-Namens schon beim Erzeugen des neuen CLI-Fensters angeben, wo es erscheinen soll und wie groß es sein soll. Es kann – zumindest für fortgeschrittene Benutzer – durchaus auch interessant sein, ein CLI auch für andere Geräte zu erzeugen. So ist es möglich, CON: durch RAW: zu ersetzen (was nur sehr selten sinnvoll ist); es kann sogar sinnvoll sein, *newcli SER:* einzugeben, wenn dieses neue CLI von einem anderen Computer aus gesteuert werden soll.

Das Gerät *RAW:* verhält sich im wesentlichen gleich wie *CON:*, reagiert aber auf Steuerzeichen in besonderer Weise. *RAW:* ist eigentlich nur für Programmierer interessant und soll deshalb hier nicht weiter besprochen werden. Wenn Sie eine experimentierfreudige Natur besitzen, können Sie die Eigenschaften von *RAW:* selbst herausfinden, indem Sie in den folgenden Beispielen immer *CON:* durch *RAW:* ersetzen. Eine Warnung aber vorweg: Wenn Sie nicht sehr vorsichtig dabei sind, wird es bei solchen Experimenten meist nötig werden, den Amiga neu zu starten!

Als Parameter hinter *newcli* genügt *CON:* allein aber nicht. Dahinter muß noch eine Beschreibung des Fensters folgen, in dem das CLI arbeiten soll. Der folgende Befehl öffnet zum Beispiel ein neues CLI-Fenster mit dem Namen *Fenster-B*, das oben links am Bildschirm erscheint und 400 Punkte breit sowie 100 Punkte hoch ist.

```
1>newcli CON:20/10/400/100/Fenster-B
```

Die allgemeine Form für die Angabe eines CON:-Gerätes lautet:

```
CON:X/Y/Breite/Höhe/Fenster-Titel
```

Dabei ist *X* der Abstand des linken Fensterrandes vom linken Bildschirmrand, *Y* der Abstand vom oberen Bildschirmrand, *Breite* die Breite des entstehenden Fensters, *Höhe* entsprechend

die Höhe und *Fenster-Titel* der Name, der oben links im neuen Fenster erscheinen soll. Enthält dieser Name Leerzeichen, so muß der gesamte Gerätename mit Gänsefüßchen eingerahmt werden. Das Gerät mit der Bezeichnung *»"CON:10/10/100/100/der Text"«* ist ein Beispiel dafür, wie dies aussehen könnte.

Sie können dieses Fenstergerät aber nicht nur im Zusammenhang mit *newcli* verwenden. Genauso, wie Sie eine Datei zum Drucker schicken, können Sie sie auch in einem neuen Fenster ausgeben lassen, indem Sie eingeben:

```
1>copy text TO CON:20/10/400/100/Fenster-B
```

Wichtig ist bei der Verwendung von CON: vor allem, daß dabei immer ein **neues Fenster** geöffnet wird, dessen Eigenschaften man ja auch angeben muß. Dies kann Vorteile haben, wie man vielleicht beim *newcli*-Befehl gesehen hat. Man kann zum Beispiel auch den Inhalt einer Datei in einem neuen Fenster zeigen, wobei der Inhalt des CLI-Fensters unbeeinträchtigt bleibt. Dies Verhalten ist aber nicht unbedingt immer das, was man möchte.

Will man, daß ein Befehl dasselbe Fenster für Ein- oder Ausgaben nimmt, in dem man selbst gerade mit dem CLI arbeitet, so darf man nicht das Gerät CON: verwenden. Das aktuelle CLI-Fenster wird immer durch das Zeichen * (Stern) symbolisiert. Nehmen wir das Beispiel von oben, bei dem eine Textdatei am Bildschirm ausgegeben wird. Soll dies nicht in einem neuen, sondern im aktuellen Fenster geschehen, so müßte der Befehl lauten:

```
1>copy text TO *
```

7.9.2 Hintergrund-Befehle

Eine vielleicht noch praktischere Art, das Multitasking des Amiga zu nutzen, als mehrere CLIs zu verwenden, sind die »Hintergrund-Befehle«. Wenn Sie einen Befehl aufrufen wollen, von dem Sie genau wissen, daß er zwar lange dauert, aber keine weiteren Tastatureingaben mehr benötigt und nur wenig Bildschirmausgaben macht, so brauchen Sie dazu kein neues CLI zu öffnen. Schließlich benötigt jedes zusätzliche Fenster zusätzlichen Speicherbedarf. Sie können statt dessen jedes CLI-Kommando mit dem Befehl *run* (engl. für *lauf los*) »im Hintergrund« laufen lassen.

Den Befehl *run* können Sie an den Anfang jeder Befehlszeile im CLI schreiben und er führt den dahinter folgenden Befehl als separaten Task parallel zum CLI aus. Lassen Sie beispielsweise einmal den *dir*-Befehl im Hintergrund laufen:

```
1>run dir c:
```

Das Prompt erscheint dann sofort wieder und Sie könnten auch sofort einen neuen Befehl eingeben. Zur Kommunikation mit Ihnen verwendet aber auch dieses im Hintergrund laufende Programm dasselbe Fenster wie das CLI im Vordergrund. Läuft mehr als ein Programm, das Ein- und Ausgaben im Fenster machen will, kann dies sehr verwirrend wirken. Falls Sie einen Eindruck von dem Chaos haben wollen, das dann am Bildschirm entstehen kann, geben Sie einmal die beiden folgenden Befehle ein:

```
1>run dir c:
1>dir c:
```

Für die Hintergrund-Verarbeitung eignen sich wirklich nur die Befehle, die keine weitere Interaktion mit dem Anwender benötigen.

Wenn Sie genau wissen, daß Sie mehrere Befehle hintereinander aufrufen wollen, so können Sie alle auf einmal hinter *run* eingeben. Nachdem Sie den ersten Befehl komplett eingegeben haben, tippen Sie dazu ein '+' an das Ende der Befehlszeile und dann erst [Return]. Die Schreibmarke geht daraufhin in die nächste Zeile. Der Befehl in der gerade abgeschlossenen Zeile wird aber nicht sofort ausgeführt! Das Plus-Zeichen ist ein Signal für das CLI, zu warten, da noch weitere Befehle folgen. Es bewirkt praktisch, daß die erste Zeile »logisch verlängert« wird, die zweite Zeile also für das CLI gar keine neue, sondern eine Fortsetzung der ersten Zeile ist. Auf diese Weise können Sie nahezu beliebig viele Zeilen, die jeweils einen Befehl enthalten, zu einer »logischen Zeile« verketten.

Erst wenn Sie [Return] drücken, ohne unmittelbar davor ein Plus-Zeichen eingegeben zu haben, beginnt das CLI, alle zuvor eingegebenen Befehle nacheinander auszuführen. Einer nach dem anderen wird gesucht, geladen und aufgerufen. Leider fährt das CLI auch dann noch mit der Ausführung der restlichen Befehle fort, wenn »weiter vorne« ein Fehler aufgetreten ist. Dies macht die gleichzeitige Eingabe mehrerer Befehle zu einer etwas riskanten Sache. Angenommen, Sie hängen zwei Befehle auf diese Art hintereinander. Der erste kopiert eine Datei auf eine andere Diskette, der zweite löscht anschließend das Original. Wenn beim Kopieren der Datei ein Fehler auftritt, weil zum Beispiel die Zieldiskette schreibgeschützt oder fehlerhaft ist, fährt das CLI trotzdem fort und löscht das Original. Sie stehen dann da und haben weder ein Original noch eine Kopie. Falls es das einzige Exemplar einer wichtigen Datei war, kann wochenlange Arbeit verloren sein!

Obwohl die Verkettung beziehungsweise »Verlängerung« von Befehlszeilen mit dem Plus-Zeichen in Zusammenhang mit dem *run*-Befehl eine sehr praktische Sache ist, sollten Sie sie sehr vorsichtig anwenden – zumindest dann, wenn Befehle dabei sind, die direkt oder indirekt destruktiv wirken können! Dasselbe gilt sinngemäß natürlich auch für die Eingabe eines neuen Befehls, solange der vorangehende noch nicht beendet ist (sieheoben). Auch hierbei kontrolliert das CLI nicht, ob Fehler aufgetreten sind, sondern führt den neuen Befehl »blindgläubig« aus.

7.10 Ein ganz spezielles Gerät: die RAM-Disk

Zum Abschluß dieses Kapitels über das CLI soll noch ein Thema behandelt werden, das nicht ausschließlich zum CLI gehört. Es konnte aber bislang, ohne Grundkenntnisse über das CLI, nicht sinnvoll erörtert werden. In diesem und den vorangehenden Kapiteln war mehrfach von einer sogenannten RAM-Disk die Rede. Um diese RAM-Disk geht es hier. Die Verwendung einer RAM-Disk kann Ihre Arbeit wesentlich beschleunigen und vor allem auch Ihr Leben einfacher machen, wenn Sie kein externes Disketten- oder Festplattenlaufwerk besitzen und mit dem internen Diskettenlaufwerk des Amiga 500 auskommen müssen.

Es kann sein, daß Sie eine RAM-Disk schon einmal auf der Workbench gesehen haben. Bei einigen Versionen der Workbench-Diskette, die mit dem Amiga 500 ausgeliefert wurden, taucht gleich nach dem Start ein Disketten-Piktogramm namens *RAM-Disk* auf der Workbench auf. Wenn man dieses Piktogramm öffnet, ist es allerdings leer. Falls bei Ihnen ein solches Piktogramm nicht auftaucht, machen Sie sich keine Sorgen. Sie werden bald erfahren, wie Sie es erzeugen können.

7.10.1 Was eine RAM-Disk ist

Hinter dem Piktogramm RAM-Disk auf der Workbench und dem Gerät *RAM:* im CLI verbirgt sich eine sogenannte »virtuelle Diskette«, die man meist »RAM-Disk« nennt. Eine solche RAM-Disk ist in Wirklichkeit etwas Software im Betriebssystem, die eine Diskette im RAM-Speicher des Amiga 500 simuliert. Wenn Sie eine Datei auf diese Diskette kopieren, wird diese Datei in Wirklichkeit im RAM-Speicher abgelegt. Und wenn Sie diese Datei auf dem Bildschirm ausgeben, kommen die Daten aus dem RAM-Speicher und nicht von einer Diskette.

Da diese *RAM-Disk*ette aber keine beweglichen Teile besitzt, ist sie natürlich extrem schnell. Wenn Daten auf dieser »Diskette« geschrieben oder von ihr gelesen werden, werden sie ja nur im Speicher hin und her geschaufelt. Das ist der Hauptvorteil der RAM-Disk. Ihr Hauptnachteil ist ihre Flüchtigkeit. Wenn der Amiga 500 angeschaltet wird oder Sie einen Neustart durchführen, geht der Inhalt der RAM-Disk verloren. Wichtige Daten oder Programme, die sich ansonsten auf keiner Diskette befinden, auf die RAM-Disk zu legen, ist so gut wie versuchter »elektronischer Selbstmord«.

7.10.2 Wie Sie die RAM-Disk sichtbar machen

Die RAM-Disk ist immer vorhanden. Sie kann nur unter Umständen unsichtbar sein, das heißt, es taucht kein entsprechendes Piktogramm auf der Workbench auf. Falls bei Ihnen die RAM-Disk nicht schon auf Ihrer Workbench erscheint, müssen Sie in das CLI gehen, um ein entsprechendes Piktogramm zu erzeugen. Von der Workbench aus können Sie die RAM-Disk nicht sichtbar machen.

Das Sichtbarmachen der RAM-Disk ist aber sehr einfach. Sie brauchen dazu nicht mehr zu tun, als sie zu benutzen. Geben Sie zum Beispiel den folgenden Befehl ein und die RAM-Disk erscheint als Piktogramm in der Workbench:

```
1>copy :Disk.Info to RAM:
```

Die RAM-Disk wird nämlich auf der Workbench sichtbar, sobald sie einmal benutzt wurde. Der obengenannte *copy*-Befehl macht genau das. Er legt eine Kopie der Datei *Disk.Info*, die unter anderem die Informationen enthält, wie ein Disketten-Piktogramm aussieht auf die RAM-Disk, die im CLI unter dem Namen *RAM:* erscheint. Sie haben damit übrigens eine weitere Form des *copy*-Befehls kennengelernt, die oft auch sehr praktisch ist. Normalerweise benötigt *copy* ja zwei Parameter. Der erste (FROM) ist der Name der Datei, die kopiert werden soll, der zweite (TO) ist der Name der Kopie. Ist der zweite Name aber der Name eines Directory, so wird die Kopie in dieses Directory gelegt und erhält denselben Namen wie das

Original. Deshalb liegt nun im Directory *RAM:*, dem Wurzeldirectoy der RAM-Disk, eine Datei namens *Disk.Info*.

Sie brauchen aber nicht unbedingt eine Datei auf die RAM-Disk zu kopieren. Auch der folgende Befehl, der scheinbar nichts tut (solange die RAM-Disk) leer ist, bringt das RAM-Disk-Piktogramm auf der Workbench zum Vorschein:

```
1>dir RAM:
```

7.10.3 Wie Sie die RAM-Disk benutzen

Sofort nach dem Start können Sie die RAM-Disk im CLI verwenden wie jedes andere Laufwerk auch. Verwenden Sie dazu zum Beispiel in Kopierbefehlen nur den Laufwerksnamen *RAM:* statt *DF0:* oder *DF1:*. Auf der Workbench müssen Sie unter Umständen das RAM-Disk-Piktogramm erst sichtbar machen. Dann allerdings können Sie dieses Piktogramm verwenden wie jedes andere Disketten-Piktogramm. So ist es möglich, Piktogramme von Projekten und Werkzeugen daraufzulegen, Schubladen daraufzulegen, Programme zu starten und so weiter.

Berücksichtigen Sie aber, daß die RAM-Disk im RAM-Speicher Ihres Amiga 500 existiert. Jedes Piktogramm, das Sie auf die RAM-Disk legen, und jede Datei, die Sie nach *RAM:* kopieren, kostet Sie RAM-Speicher. Das können Sie auch direkt verfolgen, wenn Sie die Anzeige des freien Speicherplatzes in der Titelleiste der Workbench während eines Kopiervorgangs betrachten. Legen Sie deshalb keine überflüssigen Daten und Programme auf die RAM-Disk! Besonders auf dem Amiga 500 ohne Speichererweiterung kann es rasch passieren, daß Sie nicht mehr genügend Speicher für Programme freihaben, wenn Sie viele Daten auf die RAM-Disk legen.

Eine merkwürdige Eigenschaft der RAM-Disk verdient auch noch der Erwähnung: sie ist immer randvoll! Das können Sie feststellen, wenn Sie das Diskettenfenster der RAM-Disk auf der Workbench betrachten (die Füllanzeige ist immer am oberen Anschlag) und wenn Sie das Disketten-Piktogramm anklicken und dann *Info* aufrufen (die Anzahl der freien Blocks ist gleich Null). Lassen Sie sich davon aber nicht irritieren. Solange die Anzeige in der Titelleiste der Workbench noch freien RAM-Speicher meldet, ist auch auf der RAM-Disk noch Platz. Ob Sie diesen Platz allerdings völlig ausnutzen sollten, ist sehr fraglich.

7.10.4 Was Sie beachten müssen, wenn Sie die RAM-Disk benutzen

Wenn Sie die RAM-Disk benutzen wollen, müssen Sie einige Punkte beachten, um nicht böse Überraschungen zu erleben. Der wichtigste dieser Punkte ist die Tatsache, daß RAM-Disk und Programme (Werkzeuge) um den RAM-Speicher des Amiga konkurrieren. Wenn Sie also vorhaben, aufwendige große Programme zu benutzen, sollten Sie möglichst wenige Daten auf die RAM-Disk legen. Einige Programme (zum Beispiel Deluxe Paint und Deluxe Video) bieten einen erhöhten Leistungsumfang, wenn mehr RAM-Speicher zur Verfügung steht. Wenn Sie solche Programme benutzen, sollten Sie die RAM-Disk vorher leeren.

Besonders extrem ist dieses Problem auf dem Amiga 500 ohne Speichererweiterung. Hier sollten Sie nie oder nur vorübergehend mehr als etwa 200 Kbyte (400 Blocks) Daten auf die RAM-Disk legen, wenn Sie »echte« Anwendungsprogramme benutzen wollen. Großzügiger können Sie da schon arbeiten, wenn Sie die Speichererweiterung eingebaut haben. Ihnen stehen dann insgesamt 1024 Kbyte zur Verfügung. Sie können dann den Inhalt einer halben Diskette auf die RAM-Disk legen und haben immer noch mehr RAM-Speicher zur Verfügung als mit einem Amiga 500 ohne Speichererweiterung. Die meisten Programme sollten dann noch problemlos arbeiten.

Das zweite große Problem der RAM-Disk ist Ihre Flüchtigkeit. Es ist sehr gefährlich Daten auf einer RAM-Disk abzulegen, von denen man nicht eine Kopie auf Diskette hat. Sie sollten deshalb nur Programme sowie solche Dateien auf die RAM-Disk legen, von denen eine Kopie existiert. Wenn Sie zum Beispiel einen Brief eingetippt haben und nun abspeichern wollen, legen Sie ihn niemals auf die RAM-Disk.

7.10.5 Wie Sie die RAM-Disk richtig benutzen

Die RAM-Disk bietet Ihnen zwei große Vorteile: eine hohe Geschwindigkeit und ein zusätzliches Diskettenlaufwerk. Entsprechend sollten Sie sie auch nutzen. Die hohe Geschwindigkeit sollten Sie zum Beispiel dann nutzen, wenn ein Programm viele Diskettenoperationen macht. Wenn Sie zum Beispiel ein Programm benutzen, das Bilder von der Diskette einliest und sie in möglichst rascher Folge auf dem Bildschirm zeigen soll, kann es sinnvoll sein, eine Bildfolge auf die RAM-Disk zu kopieren. Von dort kann das Programm die Bilder sehr schnell einlesen und deshalb auch in sehr schneller Folge zeigen. Auch wenn ein Programm temporäre Dateien anlegt, die nach Beendigung des Programms wieder gelöscht werden, kann man die RAM-Disk oft sinnvoll einsetzen. Lesen Sie dazu das Handbuch des entsprechenden Programms und versuchen Sie herauszufinden, wo das Programm die temporären Dateien hinlegt.

Viele Programme legen temporäre Dateien in das logische Laufwerk *T:*, das beim Systemstart dem Directory *t* auf der Workbench-Diskette zugeordnet wird. Mit dem folgenden Befehl können Sie erreichen, daß diese Dateien auf die RAM-Disk gelegt werden:

```
1> assign t: RAM:
```

Die zweite wichtige Art, die RAM-Disk zu benutzen, ist es, sie als zusätzliches Diskettenlaufwerk zu verwenden. Besonders Besitzer eines Amiga 500 ohne externes Laufwerk werden über dieses »virtuelle« Diskettenlaufwerk bald sehr dankbar sein. Wollen Sie nämlich mit nur einem Diskettenlaufwerk mehrere Dateien von einer auf eine andere Diskette kopieren, so bedeutet das sehr viele lästige Diskettenwechsel – egal ob Sie im CLI oder auf der Workbench arbeiten. Kopieren Sie jedoch alle Dateien zunächst von der einen Diskette auf die RAM-Disk, wechseln dann die Disketten aus und kopieren dann die Dateien von der RAM-Disk auf die andere Diskette, kommen Sie mit ein beziehungsweise zwei Diskettenwechseln aus – zumindest dann, wenn die zu kopierenden Daten alle auf die RAM-Disk passen. Nach Abschluß des Kopiervorgangs sollten Sie die RAM-Disk aber wieder leeren, um den dadurch belegten RAM-Speicher zurückzugewinnen.

Aber nicht nur beim Kopieren können Sie ein zusätzliches Laufwerk benötigen. Wenn Sie bei der Benutzung eines Programms ständig zwei Disketten benötigen, so können Sie den Inhalt einer dieser beiden Disketten (am besten die mit den wenigsten Daten darauf) auf die RAM-Disk kopieren. (Das bringt natürlich die wenigsten Probleme mit sich, wenn Sie mit einem Amiga mit mehr als 512 Kbyte arbeiten, also eine Speichererweiterung eingebaut haben.) Sie bekommen so entweder ein Laufwerk frei oder sparen sich lästige Diskettenwechsel, falls Sie nur ein Laufwerk besitzen. Sie können auf diese Weise sogar die Workbench-Diskette selbst einsparen. Mehr zu diesem Thema erfahren Sie in einem späteren Kapitel.

Und auf die Gefahr hin, mich zu wiederholen, noch einmal die Warnung, keine Daten auf die RAM-Disk zu legen, von denen es keine identische Kopie auf Diskette oder Festplatte gibt. Es ist zum Beispiel für manche Datenbankprogramme sehr verlockend, die Datenbanken auf die RAM-Disk zu legen. Operationen, die sonst Minuten dauern, laufen dann vielleicht in Sekunden ab. Wenn Sie jedoch Änderungen am Datenbestand vornehmen, werden diese Änderungen nur im RAM-Speicher vermerkt. Ein Stromausfall oder einer jener netten kleinen Fehler, die auch dem besten Programmierer unterlaufen, können dann die Arbeit von Stunden vernichten. Nur auf einer Diskette oder Festplatte sind Ihre Daten (einigermaßen) sicher – eine RAM-Disk ist ein höchst labiles Gebilde!

7.11 Zusammenhänge zwischen CLI und Workbench

Zum Abschluß dieser kleinen Einführung in das CLI möchte ich aber auch noch ein paar Worte zum Zusammenhang zwischen Dateiverzeichnissen und Workbench-Schubladen beziehungsweise zwischen Dateien und Projekt-Piktogrammen oder Werkzeug-Piktogrammen verlieren. Jede Workbench-Schublade entspricht immer auch einem Directory, das wir auch im CLI oder von einem Programm aus ansprechen können. Jedes Projekt (Dokument) und jedes Werkzeug (Programm), das in einem Fenster der Workbench erscheint, ist immer auch eine Datei. Der Name eines Schubladen-Piktogramms ist dabei auch immer gleich dem Namen des Dateiverzeichnisses und der Name des Projekt-Piktogramms gleich dem Dateinamen.

Umgekehrt gilt diese Beziehung aber nicht. Schon auf der Standard-Workbench befinden sich eine ganze Reihe von Dateiverzeichnissen (und natürlich auch Dateien darin), die von der Workbench aus nicht zu sehen sind. Dies sind Dateien, von denen die Entwickler des Amiga meinen, daß sie der »einfache Benutzer« nicht zu sehen braucht, beziehungsweise gar nicht sehen darf, damit er nicht irgendwelchen Unsinn damit anstellt. Diese Einstellung ist zumindest nicht völlig falsch. Eine wichtige Datei, die man nicht sieht, kann man nicht löschen.

Erst wenn man sich in das CLI begibt, bekommt man alle Dateien und Dateiverzeichnisse über die entsprechenden CLI-Befehle zu sehen. Die Entwickler des Amiga waren offensichtlich der Meinung, daß Benutzer, die sich bis zum CLI vorgearbeitet haben, bereits soviel Wissen und Verantwortungsbewußtsein (und Vorsicht!) erworben haben, daß man ihnen selbst »das Innerste des Amiga ausliefern kann«.

Im CLI sehen Sie alle Dateien, die sich auf einer Diskette befinden. In der Workbench werden Dateien und Dateiverzeichnisse erst dann sichtbar, wenn es eine Datei gibt, die den gleichen Namen mit der zusätzlichen Endung ».info« trägt. Ein Piktogramm namens *Meier* taucht also erst dann in der Workbench auf, wenn es auch eine Datei namens *Meier.info* im selben Directory gibt. (Sie können das sehr leicht mit dem *dir*-Befehl feststellen.) Zu allen Namen, die aufgelistet werden und die Sie auch auf der Workbench sehen, gibt es ein Pendant mit dem angehängten ».info«. Diese zusätzliche Datei enthält unter anderem das Piktogramm der Datei und weitere Verwaltungsinformationen, die nur die Workbench benötigt. Mehr dazu erfahren Sie noch in einem späteren Kapitel.

8 Die Kommandos des CLI

Nachdem Sie im letzten Kapitel die Grundzüge des CLI und einige der wichtigsten CLI-Kommandos kennengelernt haben, erhalten Sie auf den folgenden Seiten nun eine (fast) vollständige Liste der CLI-Kommandos. Diese Liste ist insofern unvollständig, da sie nur diejenigen Befehle umfaßt, die von einem »normalen Anwender« benötigt werden. Befehle, die speziell nur für Programmierer und Systementwickler geschaffen wurden, werden nicht aufgeführt. Informationen darüber sind in den Unterlagen, die den entsprechenden Programmiersprachen beiliegen, zu finden. Eine vollständige Liste der CLI-Kommandos zusammen mit weiteren Informationen, die für Programmierer interessant sein dürften, enthält das Amiga-DOS-Handbuch (erschienen im M&T-Verlag, Nr. 90465, Haar b. München, 1987), das Sie bei jedem Buchhändler erwerben können.

8.1 Überblick über die CLI-Kommandos

Alle CLI-Kommandos sind in Wirklichkeit kleine Programme. Anders als bei anderen Computern und deren Betriebssystemen, ist nicht ein einziger Befehl in das Programm CLI »fest eingebaut«. Wenn Sie eine Zeile mit einem Befehl im CLI-Fenster eingeben, nimmt das CLI das erste Wort dieser Zeile und sucht nach einem Programm gleichen Namens. Findet es dieses Programm, so wird es in den Speicher geladen und gestartet. Der Rest der Zeile, hinter dem Namen des Programms, wird als ein oder mehrere Parameter an das Programm weitergegeben und es bleibt diesem Programm überlassen, was es damit tut.

Das CLI sucht nur an ganz bestimmten Stellen nach solchen Programmen. Zunächst einmal wird im aktuellen Directory nach einem Programm gesucht, dessen Namen gleich dem ersten Wort in der Befehlszeile ist. Wird dort keines gefunden, so werden eine Reihe von Directories im sogenannten Suchpfad überprüft. Wird auch in diesen das Programm nicht gefunden, wird schließlich noch das »logische Laufwerk« namens C: durchsucht. (Directories und logische Geräte wurden bereits im letzten Kapitel ausführlich erläutert. Bei Unklarheiten schauen Sie

bitte dort noch einmal nach. Diese beiden Begriffe werden für dieses Kapitel laufend wieder benötigt.) Wird es auch dort nicht gefunden, so gibt es eine Fehlermeldung – sonst geschieht nichts.

Das logische Gerät *C:* wird beim Start automatisch dem Directory *<Startdiskette>:c* zugewiesen. *<Startdiskette>* muß dabei natürlich gegen den Namen der Workbench-Diskette ersetzt werden, mit der Sie den Amiga gestartet haben. Diese Zuordnung kann aber mit dem Befehl *assign*, der im folgenden noch ausführlicher erklärt wird, jederzeit geändert werden. Solange sie aber nicht geändert wird, wird der Amiga Sie jedesmal, wenn Sie einen CLI-Befehl eintippen und die Startdiskette nicht mehr in einem Laufwerk steckt, dazu auffordern, die Startdiskette einzulegen. Ähnliches gilt für die anderen Directories im Suchpfad, den Sie mit dem Befehl *path*, der ebenfalls noch im Detail erläutert wird, ändern können.

Nach diesen Erklärungen dürfte es wohl klar sein, warum das CLI eigentlich nicht über einen festen Satz von Kommandos verfügt. Löscht man eines der Programme im logischen Laufwerk *C:*, so verliert man ein mögliches Kommando. Entfernt man ein Directory aus dem Suchpfad, verliert man vielleicht eine ganze Gruppe möglicher Kommandos. Und schreiben Sie selbst ein Programm und legen es in eines der Directories im Suchpfad, so wird es dadurch automatisch ein CLI-Kommando. Auch, wenn Sie mit unterschiedlichen Disketten starten, kann es sein, daß Ihnen völlig unterschiedliche Kommando-Sätze zur Verfügung stehen.

Doch damit genug der verwirrenden Möglichkeiten. Die im folgenden noch beschriebenen Kommandos stellen den »Standardsatz« von Befehlen dar, so wie er Ihnen beim Kauf des Amiga auf der Workbench-Diskette mitgeliefert wird. Solange Sie keine Modifikationen an Ihrer Workbench-Diskette und (mit dem Befehl *path*) am Suchpfad, vornehmen, stehen Ihnen alle diese Befehle zur Verfügung. Zusätzlich dazu können auch fast alle Werkzeuge, die in den Kapiteln über die Workbench beschrieben wurden, auch vom CLI aus als Kommandos verwendet werden. Diese Programme (Werkzeuge) werden hier aber nicht noch einmal beschrieben.

8.2 Format der CLI-Kommandos

Alle CLI-Kommandos verfügen über sich ähnelnde Aufruf-Formate. Am Anfang jeder Zeile, die einen CLI-Befehl enthält, steht der Name des Befehls. Dieser Name besteht immer aus einem kurzen Wort, das im allgemeinen einem (englischen) Verb gleich oder ähnlich ist. Hinter diesem Wort kann einer oder eine Reihe sogenannter Parameter folgen, die dem Befehl genau sagen, was er zu tun hat. Ein Beispiel für einen solchen Befehl mit Parametern ist das CLI-Kommando *dir System*, das wir ja bereits kennen. *Dir* ist der Name des Befehls und *System* ist ein Parameter (der Name eines Directory, dessen Inhalt aufgerufen werden soll).

Wenn Sie den Namen des Befehls falsch beziehungsweise den Namen eines Befehls eingeben, der im Augenblick im Suchpfad nicht zu finden ist, so erhalten Sie die folgende Fehlermeldung:

```
Unknown command <name>
```

Wenn Sie hingegen den Befehlsnamen richtig eingeben und bei den Parametern einen Fehler machen, erscheint die Fehlermeldung:

```
Bad arguments
```

Sie können anhand dieser beiden Fehlermeldungen recht rasch feststellen, wo der Fehler liegt. Bei falschen Parametern kann die Fehlerbehebung jedoch schwierig werden. Parameter sagen einem Befehl, mit welchen Objekten (Dateien, Directories, Texten und so weiter) er arbeiten soll und was er genau damit tun soll (falls es mehrere Möglichkeiten gibt). Diese Parameter werden in den im nächsten Abschnitt folgenden Format-Beschreibungen als Platzhalter beschrieben. Ein solcher Platzhalter ist immer ein kurzer Begriff, der in spitze Klammern gesetzt wird (zum Beispiel *<dateiname>*). Ein Platzhalter darf aber nicht wirklich eingetippt werden – wie der Name schon sagt – sondern muß gegen einen wirklichen Parameter ersetzt werden. So muß zum Beispiel *<dateiname>* gegen einen echten Dateinamen ersetzt werden.

Wie beim *dir*-Befehl zu sehen, können Parameter optional sein, das heißt, Sie **können, müssen aber nicht** angegeben werden. Wird ein optionaler Parameter nicht angegeben, geht der Befehl von einer Voreinstellung für diesen Parameter aus. Der *dir*-Befehl nimmt, wenn er keinen Parameter erhält, zum Beispiel an, daß Sie die Liste der Dateien im aktuellen Directory sehen wollen. Ist ein Parameter optional, muß er also zur Befehlsausführung nicht eingegeben werden, so wird sein Platzhalter in den folgenden Beschreibungen kursiv gedruckt.

Solange ein Befehl nur einen möglichen Parameter hat, ist die ganze Angelegenheit mit den Parametern recht einfach. Aller Text, der hinter dem Namen des Befehls folgt, ist der Parameter. Steht hinter dem Namen kein Text mehr, ist der Parameter eben leer. Kann ein Befehl aber mehrere Parameter bekommen, wird die ganze Sache etwas schwieriger. Der Befehl *copy* zum Beispiel benötigt zum Kopieren einer Datei zwei Parameter: welche Datei kopiert werden soll und wie die Kopie heißen soll.

Der Rest der Zeile hinter dem Wort *copy* selbst muß also in zwei Teile gespalten werden. Der erste ist der Name der zu kopierenden Datei, der zweite der Name der Kopie. Damit die Teilungsstelle nicht gar so willkürlich gewählt werden muß, kennt das CLI sogenannte Schlüsselwörter (engl. *keywords*). Ein Schlüsselwort leitet einen Parameter ein und trennt ihn von den vorangehenden Parametern. Der Befehl *copy* kennt zum Beispiel das Schlüsselwort TO, das den Namen einleitet, den die Kopie der Datei tragen soll. (Schlüsselwörter werden zur Unterscheidung von Parametern immer groß geschrieben. Sie können sie in der Befehlszeile aber natürlich schreiben, wie Sie wünschen.) Die folgende Zeile legt eine Kopie der Datei *Text1* unter dem Namen *Kopie* an:

```
1>copy Text1 TO Kopie
```

TO ist ein Schlüsselwort, das eindeutig den Beginn des zweiten Parameters markiert. TO ist so eindeutig, daß Sie sogar die Reihenfolge der Parameter des *copy*-Befehls umkehren können. Der folgende Befehl leistet zum Beispiel dasselbe wie der erste *copy*-Befehl oben:

```
1>copy TO Kopie Text1
```

Wenn Sie Parameter aber in der Reihenfolge angeben, die »üblich« ist, können Sie die Schlüsselwörter oft fortfallen lassen. Statt *copy Text1 TO Kopie* genügt zum Beispiel auch *copy Text1 Kopie*. Solche Schlüsselwörter, die fortfallen können, heißen optionale Schlüsselwörter. Sie werden genau wie optionale Parameter kursiv gedruckt. Trennen Sie aber bitte immer zwischen Schlüsselwort und Parameter. Wenn ein Schlüsselwort optional ist, muß es der zugehörige Parameter noch lange nicht sein. Auch dafür ist der *copy*-Befehl ein gutes Beispiel. TO ist zwar ein optionales Schlüsselwort, der zugehörige Parameter (der Name der Kopie) muß aber angegeben werden.

Wenn Sie ein (optionales) Schlüsselwort fortfallen lassen, identifiziert das CLI die Parameter anhand ihrer Reihenfolge. Sie dürfen in diesem Fall die Reihenfolge der Parameter also auf keinen Fall ändern, so wie Sie es können, wenn Sie Schlüsselwörter verwenden.

Eine besondere Kategorie der Schlüsselwörter sind die »Schalter«. Diese Schlüsselworte leiten keinen Parameter ein, sondern stehen für sich allein. Ein Beispiel für einen solchen Schalter haben Sie schon bei der Beschreibung des *delete*-Befehls im letzten Kapitel kennengelernt. Wenn Sie hinter *delete* den Namen eines Directory und dann das Schlüsselwort ALL angeben, werden alle Dateien in diesem Directory zusammen mit dem Directory selbst gelöscht. Schlüsselwörter, die als Schalter funktionieren, sind übrigens fast immer optional (und werden deshalb auch kursiv geschrieben).

Zuletzt noch ein Wort zu den unterschiedlichen Parametern, die Sie verwenden können. Meist sind diese Parameter Dateinamen, Directorynamen oder andere kurze Texte. Während Sie auf der Workbench in Datei- und Directorynamen durchaus auch Leerstellen verwenden können (und sogar sollen, um deren Lesbarkeit zu erhöhen), verursachen diese Leerstellen im CLI manchmal Schwierigkeiten. Die Leerstelle ist nämlich für das CLI normalerweise das Signal dafür, daß ein Parameter zu Ende ist. Sie können solche Schwierigkeiten vermeiden, indem Sie Parameter, die Leerstellen (und Sonderzeichen) enthalten, in Anführungszeichen setzen. Wenn Sie also zum Beispiel die Datei *Test* in die Datei *Neue Datei* kopieren wollen, können Sie das mit dem folgenden Befehl tun:

```
1>copy Test TO "Neue Datei"
```

8.3 Die wichtigsten CLI-Kommandos

Es folgt nun eine Liste, die Erläuterungen aller der CLI-Kommandos enthält, die Sie als »normaler Benutzer« wahrscheinlich benötigen werden. Der Übersichtlichkeit wegen beginnt jede Erklärung auf einer neuen Seite. Dies dient nicht dazu – wie böswillige Menschen vielleicht behaupten würden – die Seitenzahl dieses Buches zu erhöhen, sondern wirklich nur, um Ihnen die Arbeit damit möglichst leichtzumachen.

Alle Befehlsbeschreibungen in diesem Abschnitt folgen einem festen Format. Oben auf jeder Seite steht der Name des Befehls, danach folgt eine detaillierte Beschreibung des Aufruf-Formats, eine Kurzbeschreibung von Sinn und Zweck des Befehls und schließlich eine etwas ausführlichere Erläuterung zur Anwendung des Befehls.

Die Formatbeschreibung enthält alle wichtigen Angaben, die Sie zum Umgang mit dem Befehl brauchen. Parameter kommen in dieser Beschreibung als Platzhalter (in spitzen Klammern) vor. Schlüsselwörter werden groß geschrieben. Optionale Parameter und Schlüsselwörter werden kursiv gedruckt. Wenn Sie bei einem Parameter auch ein Muster verwenden können, um mehrere Dateien auf einmal zu bearbeiten (wie im letzten Kapitel beschrieben), folgt dem entsprechenden Platzhalter ein Sternchen (*).

Die Erläuterungen bleiben absichtlich knapp und nüchtern und sind nur mit wenigen kurzen Beispielen versehen. Eine ausführliche Erläuterung jedes der etwa 50 Befehle mit Beispielen könnte nämlich alleine schon ein Buch füllen. Mit ein wenig Experimentierfreude werden Sie aber auch die komplexeren Befehle mit Sicherheit rasch anwenden können! Die folgende Beschreibung des *copy*-Befehls soll Ihnen als Beispiel dafür dienen, wie eine solche Befehlsbeschreibung aussieht:

copy

Format: copy *FROM* <name1>* *TO* <name2> *ALL QUIET*

Zweck: dient zum Kopieren einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

Erläuterung: ...

Diese Beschreibung sagt aus, daß *copy* zwei unbedingt notwendige Parameter <name1> und <name2> besitzt. Für <name1> können Sie auch ein Muster verwenden. Dies ist am Sternchen hinter <name1> zu erkennen. Die beiden Parameter können von den optionalen (kursiv gedruckten) Schlüsselwörtern FROM und TO eingeleitet werden. Zusätzlich dazu gibt es noch zwei Schlüsselwörter, die als Schalter wirken: ALL und QUIET, die ebenfalls optional sind (da kursiv gedruckt).

Dieses Format der Befehlsbeschreibungen soll zwei sehr unterschiedliche Benutzungsweisen des Kapitels erleichtern. Einmal das sequentielle Lesen und Experimentieren, mit dem Sie ja hoffentlich gleich beginnen werden. Gleichzeitig sollen die folgenden Beschreibungen aber auch als Nachschlagewerk für den täglichen Gebrauch des CLI dienen. Hierfür dürften oft die Beschreibungen des Aufruf-Formats der einzelnen Kommandos genügen.

;

Format: <Befehl> ; <Kommentar>

Zweck: dient der Möglichkeit, erläuternde Kommentare hinter einen CLI-Befehl anzufügen.

Erläuterung: Das Semikolon ist kein richtiger Befehl, sondern ein Sonderzeichen, das, wenn es in einer Befehlszeile auftaucht, vom CLI besonders interpretiert wird. Es ist ein Signal für das CLI, alles, was dahinter noch steht, zu ignorieren. Schreibt man hinter einen kompletten CLI-Befehl ein Semikolon und danach einen Text, so hat dieser Text keine Wirkung. Auf diese Weise kann der entsprechende Befehl mit einem Kommentar versehen werden. Dies ist nicht besonders wichtig, wenn man den Befehl gerade eintippt – schließlich weiß man in diesem Augenblick, was man damit bezweckt. Es wird aber sehr wichtig, wenn man eine Folge von Befehlen zu einer Kommandofolge zusammenfaßt. Mehr über Kommandofolgen in einem weiteren Kapitel und bei der Erläuterung des Befehls *execute*.

addbuffers

Format: addbuffers *DRIVE* <Laufwerk> *BUFFERS* *<N>*

Zweck: dient zur Beschleunigung der Arbeit mit einem Diskettenlaufwerk oder einer Festplatte.

Erläuterung: Mit dem Befehl *addbuffers* werden dem Diskettenlaufwerk <Laufwerk> <N> Blöcke à 512 Bytes als zusätzlicher Pufferspeicher zugewiesen. In diesen Pufferspeicher werden alle Daten gelegt, die von der Diskette gelesen werden. Werden diese Daten ein zweites Mal benötigt, werden sie nicht mehr von der Diskette geholt, sondern aus dem Puffer, was ungleich schneller geht. Je größer der Puffer (also <N>), desto spürbarer wird die Beschleunigung der Arbeit.

Der Pufferspeicher geht aber natürlich als Arbeitsspeicher verloren, reduziert also den Platz im RAM-Speicher, der Programmen und Ihren Daten zur Verfügung steht. Weisen Sie nur solchen Disketten mit *addbuffers* zusätzlichen Pufferspeicher zu, von denen Sie genau wissen, daß Sie häufig darauf zugreifen. Und schließlich: weisen Sie niemals der RAM-Disk Pufferspeicher zu. Das wäre echte Verschwendung, da diese »Diskette« ja schon im RAM liegt.

addbuffers wird typischerweise in der Startup-Sequence verwendet, einer Folge von CLI-Kommandos, die automatisch bei jedem Neustart durchlaufen wird.

Ein Beispiel:

```
1>addbuffers DRIVE df0: BUFFERS 20
```

Dieser Befehl beschleunigt das Arbeiten mit der Diskette im Laufwerk df0:, wodurch Sie allerdings auch 10 Kbyte RAM-Speicher verlieren. Dasselbe leistet der folgende Befehl – falls das logische Laufwerk *SYS:* im Laufwerk *df0:* liegt:

```
1>addbuffers SYS: 20
```

assign

Format: assign *NAME <Name> DIR <Dir>*

Zweck: dient zum Erzeugen oder Ändern logischer Geräte.

Erläuterung: Mit dem Befehl *assign* ist es möglich, die Zuordnung von logischen Geräten (logischen Laufwerken) zu Dateien oder Directories zu ändern.

Rufen Sie *assign* ohne Parameter auf, wird unter der Überschrift Directories eine Liste aller logischen Geräte und der Directories, denen sie entsprechen, angezeigt. Außerdem werden unter der Überschrift *Devices* (engl. für *Geräte*) auch noch die wirklichen »physikalischen« Geräte aufgelistet, die Sie verwenden können.

Rufen Sie nur *assign* auf und geben nur den ersten Parameter <Name> an, so wird dieses logische Gerät gelöscht. Dies ist sehr gefährlich bei den standardmäßig definierten logischen Geräten! Bei diesen sollte niemals versucht werden, die Zuordnungen zu löschen!

Der »Normalfall« ist aber ein Aufruf von *assign* mit zwei Parametern. Der erste ist der Name eines logischen Gerätes, der zweite der Name einer Datei oder meist eines Directory, das mit diesem logischen Gerät verbunden werden soll. Der Name eines logischen Gerätes sollte immer mit einem Doppelpunkt enden! Handelt es sich um ein schon vorhandenes logisches Gerät, so wird die Zuordnung geändert. Ansonsten definiert *assign* ein neues logisches Gerät, mit dem man sich leicht Tipparbeit ersparen kann.

Ein Beispiel:

```
1>assign NAME akt: DIR df0:markus/texte/neu
1>list akt:
```

Dies ist eine praktische Befehlsfolge, wenn man Dateien aus dem Directory *df0:markus/texte/neu* häufig braucht, aber dieses Verzeichnis nicht zum aktuellen Directory machen kann. Man kann es dann mit dem Kurznamen *akt:* ansprechen.

```
1>copy C: RAM: ALL
1>assign C: RAM:
```

Diese beiden Befehle führen dazu, daß zunächst alle CLI-Befehle aus dem Directory C: auf die RAM-Disk kopiert werden und danach das logische Gerät C: auf die RAM-Disk verweist. CLI-Kommandos werden dann auf der RAM-Disk gesucht und arbeiten sehr schnell.

binddrivers

Format: binddrivers

Zweck: dient zur Aktivierung zusätzlicher Gerätetreiber, die sich im Directory *Expansion* befinden.

Erläuterung: Der Befehl *binddrivers* besitzt keine Parameter. Er durchsucht bei Aufruf das Directory Expansion auf der Workbench-Diskette nach zusätzlichen Gerätetreibern, lädt eventuell vorhandende Treiber-Dateien und aktiviert die entsprechenden Treiber. Danach sollte in der Geräte-Liste, die *assign* anzeigt, ein neues Gerät auftauchen. Zusätzliche Treiber werden üblicherweise auf Diskette mitgeliefert, wenn Sie ein Zusatzgerät (zum Beispiel eine Festplatte) kaufen, das nicht sofort von der Standard-Systemsoftware erkannt wird. Sie müssen diese Treiberdatei dann von dieser Diskette in die Schublade Expansion auf der Workbench-Diskette legen. Beim nächsten Neustart wird der Treiber dann aktiv.

binddrivers wird typischerweise in der Startup-Sequence verwendet, einer Folge von CLI-Kommandos, die automatisch bei jedem Neustart durchlaufen werden.

break

Format: break *TASK* <N> *ALL C D E F*

Zweck: dient zum Abbrechen eines als separater Task laufenden Programms

Erläuterung: Mit dem Befehl *break* können Sie ein Programm, das in einem anderen CLI-Fenster oder als Hintergrund-Task läuft, abbrechen. Hierzu müssen Sie mit dem Befehl *status* feststellen, welche »Task-Nummer« dieses Programm hat (siehe unten). Mit dieser Nummer können Sie diesen anderen Task dann abbrechen, als hätten Sie in dem entsprechenden Fenster [Ctrl]+[C] oder eine andere Tastenkombination zum Abbruch gedrückt. Wenn Sie keines der vier Schalt-Schlüsselwörter C, D, E und F verwenden, wirkt das wie die Betätigung von [Ctrl]+[C]. Verwenden Sie das Schlüsselwort ALL, werden alle Abbruch-Bedingungen (C, D, E und F) gesetzt.

Ein Beispiel:

```
1>break TASK 2 D
```

Dieser Befehl wirkt, als hätten Sie in dem Fenster, in dem Task 2 läuft (zum Beispiel dem CLI-Fenster mit der Nummer 2) [Ctrl]+[D] gedrückt. Eine laufende Kommandofolge darin wird dann abgebrochen, sobald das gerade laufende Programm beendet ist.

cd

Format: cd <*dir*>

Zweck: dient zum Ändern des aktuellen Directories und zur Feststellung, welches Directory gerade »aktuell« ist.

Erläuterung: Um dem Anwender Tipparbeit zu ersparen, kennt der Amiga das Konzept eines aktuellen Directory. Tippt man einen »einfachen« Dateinamen, der nicht mit einer Gerätebezeichnung (zum Beispiel DF0:, RAM: oder C:) beginnt, so hängt das CLI intern immer den Namen des aktuellen Directory davor, um den vollständigen Pfadnamen der Datei zu erhalten. Hat man beispielsweise eine Datei mit dem kompletten Namen *DF1:alpha/text/brief1* und das aktuelle Directory heißt *DF1:alpha/text,* so braucht man nur *brief1* zu tippen, um die Datei genau zu bezeichnen. Der Befehl *cd* (für change current directory oder ändere das aktuelle Directory) ändert den Namen, den das CLI als aktuelles Directory auffaßt.

Geben Sie *cd* ohne Parameter ein, so wird der Name des aktuellen Directory ausgegeben.

Geben Sie *cd* und dahinter den Namen eines Directories ein, so wird dieses das aktuelle Directory. Dabei muß man beachten, daß das CLI auch vor diesen Namen natürlich erst das jetzt noch aktuelle Directory setzt. Will man übergeordnete oder gar auf anderen Disketten liegende Directories »aktuell machen«, muß man den Namen des neuen Directory mit einer Laufwerksbezeichnung beginnen oder andere »Tricks« anwenden, die ausführlich im vorangehenden Kapitel erklärt worden sind.

Beachten Sie bitte, daß *cd* auch das aktuelle Laufwerk ändern kann beziehungsweise automatisch mitändert. Das aktuelle Laufwerk (das mit *:* abgekürzt werden kann) ist immer das Laufwerk, auf dem sich das aktuelle Directory befindet.

Beispiele:

```
1>cd
1>cd Briefe
1>cd df1:Texte/Briefe
```

Der erste Befehl gibt das aktuelle Directory aus, der zweite macht das Directory *Briefe* im aktuellen Directory zum neuen aktuellen Directory und der dritte Befehl macht das Directory *Briefe* im Directory *Texte* auf der Diskette im externen Laufwerk zum neuen aktuellen Directory und damit eventuell auch das externe Laufwerk zum neuen aktuellen Laufwerk.

changeTaskPri

Format: changeTaskPri *PRI* <N>

Zweck: dient zum Ändern der Priorität des CLI-Tasks.

Erläuterung: Einer der großen Vorteile des Amiga 500 ist ja seine Multitaskingfähigkeit. Mehrere Programme (Tasks) können scheinbar gleichzeitig ablaufen und völlig unterschiedliche Dinge erledigen. Jeder Task hat dabei eine Priorität, die bestimmt, welchen »Anteil« der Rechenleistung des Amiga 500 er erhält. Diese Prioritäten – und damit der Parameter <N> – können zwischen -127 und +128 liegen.

Wenn Sie in einem CLI *changeTaskPri* aufrufen, ändern Sie damit die Priorität des CLI-Tasks, von dem aus der Befehl aufgerufen wurde, und aller Tasks, die von diesem CLI aus gestartet werden. Es kann zum Beispiel sinnvoll sein, die Priorität vor dem Start eines langwierigen Hintergrundtasks (der mit *run* gestartet wird) herab- und dann wieder heraufzusetzen. Der Hintergrundtask läuft dann zwar recht langsam, stört Sie aber nicht beim Weiterarbeiten im CLI.

Verwenden Sie für <N> bitte keine Prioritäten außerhalb des Bereichs von -5 bis +5, außer Sie wissen genau, was Sie tun. Prioritäten außerhalb dieses Bereichs kommen normalerweise nämlich der Arbeit des Betriebssystems Amiga-DOS selbst »ins Gehege«.

copy

Format: copy *FROM* <name1>* *TO* <name2> *ALL QUIET*

Zweck: dient zum Kopieren einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

Erläuterung: Die meistverwendete Möglichkeit des *copy*-Befehls ist sicherlich, einzelne Dateien oder die Inhalte kompletter Directories zu kopieren.

Ist sowohl <name1> als auch <name2> ein Dateiname (also nicht der Name eines schon existierenden Directory), wird der komplette Inhalt der Datei <name1> in die Datei namens <name2> kopiert. Existierte <name2> schon, geht der alte Inhalt von <name2> verloren! Existierte er noch nicht, so wird eine neue Datei dieses Namens angelegt. <name2> kann auch der Name eines Gerätes sein. Kopiert man beispielsweise eine Datei zum Gerät PRT: (dem Drucker), so bewirkt dies, daß die entsprechende Datei ausgedruckt wird.

Sind <name1> und <name2> Namen von Directories, so werden alle Dateien, die sich im Directory <name1> befinden, in das Directory <name2> kopiert (das dazu schon existieren muß). Wollen Sie auch Directories, die sich in <name1> befinden, und deren Inhalt kopieren, müssen Sie an das Ende des Befehls noch das Schlüsselwort ALL schreiben. Ohne ALL werden nur Dateien kopiert, die sich in <name1> befinden, weitere Directories in diesem bleiben unberücksichtigt.

Wenn auf diese Weise mehrere Dateien auf einmal durch einen *copy*-Befehl kopiert werden, protokolliert *copy* diesen Vorgang, indem immer der Name der gerade bearbeiteten Datei im CLI-Fenster ausgegeben wird. Man hat so einen stetigen Überblick darüber, wie weit der Kopiervorgang schon ist. Selbst wenn der Vorgang wegen eines Fehlers (wenn zum Beispiel die Zieldiskette voll ist) abbricht, weiß man genau, bei welcher Datei dies geschehen ist. Will man – aus welchen Gründen auch immer – dieses Protokoll nicht sehen, so kann man das Schlüsselwort QUIET (engl. für *still* oder *ruhig*) an das Ende des Befehls schreiben. Dann unterbleiben alle Ausgaben während des Kopiervorgangs.

Eine letzte Möglichkeit, den *copy*-Befehl zu verwenden, besteht darin, statt <name1> kein Directory, sondern ein Muster anzugeben, in das die Dateinamen passen müssen, um kopiert zu werden. <name2> muß dazu aber ein Directory sein. Alle Dateien, die in das Muster <name1> passen, werden dann in das Directory <name2> kopiert.

Beispiele:

```
1>copy Brief TO KopieBrief
1>copy Brief TO "Kopie von Brief"
1>copy Brief TO :Texte/alteBriefe
1>copy :Texte/Briefe TO :Texte/alteBriefe
1>copy :Texte/Briefe TO :Texte/alteBriefe ALL
1>copy #?.BAK TO df0:bak
```

Der erste Befehl legt eine Kopie der Datei *Brief* im aktuellen Directory unter dem Namen *KopieBrief* an. Der zweite verwendet für diese Kopie den Namen *Kopie von Brief* (mit Leerstellen). Der dritte Befehl kopiert *Brief* (unter demselben Namen) in das Directory *:Texte/alteBriefe*. Der vierte Befehl kopiert alle Dateien (aber nicht Directories und deren Inhalt) aus dem Directory *:Texte/Briefe* in das Directory *:Texte/alteBriefe*. Der fünfte Befehl kopiert den **kompletten** Inhalt von *:Texte/Briefe* in das Directory *:Texte/alteBriefe*. Und der letze Befehl kopiert alle Dateien im aktuellen Directory, die mit ».BAK« enden, in das Directory *bak* auf der Diskette im internen Laufwerk.

date

Format: date *TIME <zeit> DATE <datum> TO <datei>*

Zweck: dient zum Anzeigen oder Ändern der vom System gespeicherten Zeit und des Datums.

Erläuterung: Der Befehl *date* wird wohl am häufigsten dafür verwendet, die Systemzeit nachzustellen. In der Grundausbaustufe hat der Amiga 500 ja keine eingebaute, batteriegespeiste Uhr – diese ist erst in der Speichererweiterung Amiga 501 enthalten. Deshalb wird dieses Nachstellen nach jedem Neustart wieder nötig. Beim Start liest der Amiga zunächst einmal das Datum und die Uhrzeit der letzten Modifikation an der Startdiskette und verwendet dieses als aktuelle Zeit. Wer Wert darauf legt, daß die Datumsangaben, die ihm zum Beispiel bei Anwendung des *list*-Befehls (siehe unten) gezeigt werden, wirklich stimmen, der sollte deshalb nach jedem Start die Uhr nachstellen. Die Verwendung des CLI-Befehls *date* für diesen Zweck ist meist wesentlich schneller als die Einstellung der Zeit mittels *Preferences*, wie es innerhalb der Workbench üblicherweise gemacht wird. Wer also das CLI benutzt, sollte nicht den Umweg über *Preferences* wählen.

Mittels *date* können sowohl Datum als auch Uhrzeit getrennt oder in einem Zug eingestellt werden. Hierzu dienen die optionalen Parameter <datum> und <zeit>, die von den beiden Schlüsselwörtern DATE und TIME eingeleitet werden können. Das Datum sollte dazu im Format DD-MMM-JJ (zum Beispiel 07-Jul-87) und die Uhrzeit im Format HH:MM:SS (zum Beispiel 11:44:00; die Sekundenangabe kann auch fortfallen). Die Reihenfolge, in der Uhrzeit und Datum eingegeben werden, spielt keine Rolle – auch dann nicht, wenn Sie die Schlüsselwörter DATE und TIME fortfallen lassen, *Date* erkennt die Uhrzeit am Doppelpunkt, das Datum am Bindestrich.

Besonders für die Besitzer eines Amiga 500 ohne Speichererweiterung und Uhr sind einige sehr spezielle »Datumsangaben« interessant, die *date* außer den üblichen Daten auch noch versteht: die (englischen) Namen der Wochentage und *tomorrow* (engl. für *morgen*) sowie *yesterday* (engl. für *gestern*). Sie können diese Wörter statt eines Datums wie 01-jan-87 verwenden. Die Angaben *tomorrow* und *yesterday* setzen das Datum jeweils einen Tag vor oder zurück und die Angabe eines Wochentagsnamens setzt das Datum auf den nächsten Wochentag mit diesem Namen.

Wenn Sie *date* ohne Parameter eingeben, werden aktuelle Uhrzeit und aktuelles Datum (oder jedenfalls das, was Ihr Amiga 500 dafür hält) im CLI-Fenster ausgegeben. Wollen Sie diese Ausgabe auf eine Datei oder ein Gerät umlegen, so können Sie dessen Namen hinter dem Schlüsselwort TO angeben.

Beispiele:

```
1>date
1>date TO datum
1>date TO PRT:
1>date TIME 11:40
1>date 07-jul-87 11:40:30
```

Der erste Befehl gibt Datum und Uhrzeit auf dem Bildschirm aus. Der zweite und dritte schicken diese Informationen in die Datei *datum* beziehungsweise auf den Drucker. Der vierte Befehl stellt die Zeit neu ein (auf 11 Uhr 40). Und der fünfte Befehl stellt Datum und Uhrzeit (auf den siebten Juli 1987, 11 Uhr 40 und 30 Sekunden). Die folgenden Beispiele illustrieren noch einmal die Verwendung von Wochentagsnamen.

```
1>date tomorrow
1>date yesterday
1>date wednesday
```

Der erste Befehl setzt das Datum einen Tag weiter, der zweite einen Tag zurück. Und der dritte Befehl setzt das Datum auf den nächsten Tag, der ein Mittwoch (engl. *wednesday*) ist.

delete

Format: delete <name1>* *<name2> ... <name10> ALL QUIET Q*

Zweck: dient zum Löschen einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

Erläuterung: Das Kommando *delete* dient zwar zum Löschen von Dateien, ist ansonsten dem copy-Befehl vom Format her aber recht ähnlich. Wenn Sie einzelne Dateien löschen wollen, so können Sie deren Namen (bis zu zehn Stück) direkt hinter *delete* als Parameter <name1> bis <name10> angeben. Alle oder einige dieser Namen können auch Directories sein. Diese müssen allerdings leer sein, um gelöscht zu werden.

Geben Sie hingegen nur einen Directorynamen als Parameter an und schreiben an das Ende der Befehlszeile noch das Schlüsselwort ALL, so werden auch alle Dateien und Directories innerhalb von <name1> und schließlich auch das Directory <name1> selbst gelöscht. Dabei wird immer der Name der Datei ausgegeben, die gerade gelöscht wird. Wenn Sie diese Auflistung unterdrücken wollen, müssen Sie hinter ALL noch das Schlüsselwort QUIET oder kurz Q schreiben.

Statt für <name1> den Namen eines Directory anzugeben, können Sie an dieser Stelle auch ein Muster angeben. Alle Dateien, deren Namen in dieses Muster passen, werden dann gelöscht. Auch hier können Sie mit dem Schlüsselwort QUIET oder Q die Auflistung der gelöschten Dateien während des Löschvorgangs verhindern. Wie solche Muster genau aussehen können, wurde im vorangegangenen Kapitel detailliert beschrieben.

Beispiele:

```
1>delete Brief1 Brief2 Brief3
1>delete :Texte/alteBriefe ALL
1>delete #?.BAK
```

Der erste dieser Befehle löscht die drei Dateien *Brief1*, *Brief2* und *Brief3*. Der zweite löscht das Directory *:Texte/alteBriefe* und der letzte *delete*-Befehl schließlich löscht alle Dateien im aktuellen Directory, die mit ».BAK« enden.

dir

Format: dir *<dirname>* OPT A OPT I OPT AI OPT D*

Zweck: dient zum Anzeigen des Inhalts von ganzen Disketten und Directories.

Erläuterung: Der Befehl *dir* ist wahrscheinlich der am häufigsten benötigte CLI-Befehl. Er ist die wichtigste Möglichkeit, mit der man einen Überblick über den Inhalt von Directories gewinnen kann. Detailliertere Informationen über einzelne Dateien lassen sich mit dem Befehl *list* gewinnen.

Gibt man den Befehl *dir* ohne einen Parameter dahinter ein, so wird der Inhalt des aktuellen Directories aufgelistet. Folgt auf *dir* der Name eines Directory, so wird dessen Inhalt aufgelistet. Dabei werden zunächst die weiteren Directory, die sich in dem jeweiligen Directory befinden, und dann die Dateien in alphabetisch sortierter Form gezeigt. Die Liste der Directories ist einspaltig und hinter jedem Namen steht noch einmal in Klammern der Hinweis »dir«. Die Auflistung der Dateien erfolgt zweispaltig.

Das Schlüsselwort OPT erlaubt eine weitere Modifikation der Arbeitsweise des *dir*-Befehls. Hinter OPT kann eine Reihe von Buchstaben folgen, die die gewählte **Opt**ion bestimmen. Die Option OPT A bewirkt, daß *dir* nicht nur die Namen der untergeordneten Directories auflistet, die in den ausgegebenen Directories liegen, sondern auch deren Inhalte anzeigt. Man kann sich damit beispielsweise eine komplette Liste **aller** Dateien auf einer Diskette ausgeben lassen. Jedesmal, wenn *dir* bei dieser Auflistung eine Stufe tiefer in den Dateienbaum geht, werden die Namen etwas nach rechts eingerückt.

Die Option OPT D bewirkt, daß nur Directories und keine Dateien aufgelistet werden. Die Option OPT I erlaubt ein »interaktives Durchsuchen« von Directories, das im folgenden noch näher erklärt wird. Und die Option OPT AI ist eine Kombination von OPT A und OPT I.

Wird bei einem *dir*-Befehl die I-Option (I für ***i**nteraktiv*) gewählt, so wird Ihnen zunächst jeder Name eines Directory oder einer Datei einzeln gezeigt und dann ein Fragezeichen ausgegeben. Sie können nun durch die Eingabe eines Buchstabens bestimmen, wie weiter vorzugehen ist:

Drücken Sie die [Return]-Taste, so wird Ihnen der nächste Name gezeigt. Tippen Sie Q und dann [Return], so wird der *dir*-Befehl abgebrochen. Dasselbe können Sie mit [Ctrl]+[C] erreichen. Wird der Name eines Directory angezeigt und Sie tippen [E] und dann [Return], wird der Inhalt dieses Directory bearbeitet. Der nächste Name, der dann erscheint, ist der erste Name aus diesem untergeordneten Directory. Befinden Sie sich in einem untergeordneten Directory und drücken Sie die Taste [B] (für *back = zurück*) und dann [Return], so wird im darüberliegenden Directory an der Stelle weitergemacht, wo Sie zuvor den E-Befehl gegeben haben. Mit E und B kann man sich auf diese Weise durch den Baum der Directories »hangeln«.

Tippt man hinter dem Fragezeichen »DEL« (nicht die [Del]-Taste, sondern die Buchstaben D, E und L), so wird die gerade angezeigte Datei gelöscht. Auch Directories können auf diese Weise gelöscht werden, aber nur, wenn sie zuvor vollständig geleert wurden (durch Löschen aller Dateien und Directories in ihnen). Wird der Name einer Datei angezeigt und man tippt

[T] und dann [Return], so wird der Inhalt dieser Datei im CLI-Fenster ausgegeben. (Dies ist natürlich nur dann sinnvoll, wenn es sich beim Inhalt dieser Datei um einen Text handelt. Ansonsten können unvorhersehbare Schwierigkeiten auftauchen.) Haben Sie genug vom Dateiinhalt gesehen und möchten die Ausgabe deshalb abbrechen, brauchen Sie nur mit [Ctrl]+[C] abzubrechen.

Sind Sie sich während eines interaktiven *dir*-Befehls im unklaren darüber, welche Möglichkeiten Sie gerade haben, tippen Sie hinter dem Fragezeichen einfach noch ein Fragezeichen. Eine Liste der im Augenblick möglichen Eingaben wird Ihnen dann angezeigt.

Beispiele:

```
1>dir Briefe
1>dir :Texte/alteBriefe
1>dir df0: OPT A
1>dir df0: OPT I
1>dir #?.BAK
```

Der erste dieser Befehle listet den Inhalt des Directory *Briefe* im aktuellen Directory auf, während der zweite das Directory *Texte/alteBriefe* auf der aktuellen Diskette auflistet. Der dritte Befehl zeigt den kompletten Inhalt der Diskette im internen Laufwerk. Der vierte leitet eine interaktive Durchsuchung der Directories und Dateien im internen Laufwerk ein und der letzte *dir*-Befehl schließlich zeigt alle Dateien im aktuellen Directory, die mit ».BAK« enden.

diskchange

Format: diskchange *DEV* <laufwerkname>

Zweck: dient zum Anmelden eines Diskettenwechsels.

Erläuterung: Der Befehl *diskchange* teilt Amiga-DOS mit, daß die Diskette im Laufwerk <laufwerkname> (zum Beispiel *df1:*) ausgewechselt wurde. Bei den 3½-Zoll-Laufwerken von Commodore ist dieser Befehl nicht notwendig. Bei diesen merkt Amiga-DOS von selbst, wenn eine Diskette herausgenommen oder eingelegt wird. Bei einigen 5¼-Zoll-Laufwerken kann dieser Befehl nötig sein und ist dann enorm lästig). Kaufen Sie deshalb am besten kein Laufwerk, das einen Diskettenwechsel nicht von selbst (hardwaremäßig) Amiga-DOS mitteilt.

diskcopy

Format: diskcopy *FROM* <diskname1> TO <diskname2> *NAME* *<name>*

Zweck: dient zum Kopieren kompletter Disketten (vor allem zur Erstellung von Sicherungskopien).

Erläuterung: Der Befehl *diskcopy* legt eine komplette Kopie einer Diskette auf einer anderen Diskette an. Der alte Inhalt der zweiten Diskette wird dabei vollständig zerstört. *Diskcopy* ist besonders für das regelmäßige Anlegen von Sicherungskopien sehr sinnvoll. Bei Anwendung von *diskcopy* ist ein vorheriges Formatieren der Diskette (zum Beispiel mit dem CLI-Befehl *format*) nicht nötig! Für die Kopie kann deshalb sowohl eine schon verwendete als auch eine ganz neue, unbenutzte Diskette verwendet werden.

Genau wie beim Diskettenkopieren in der Workbench kann auch *diskcopy* den Kopiervorgang entweder mit einem oder zwei Diskettenlaufwerken bewältigen. So kopiert *diskcopy df0: TO df1:* zum Beispiel den kompletten Inhalt der Diskette, die sich in diesem Moment im internen Laufwerk befindet, auf die Diskette, die sich in diesem Moment im externen Laufwerk befindet. Dies dürfte die übliche Vorgehensweise sein, wenn Sie stolzer Besitzer eines externen Laufwerks sind. Bevor der Kopiervorgang wirklich beginnt, fordert der Befehl aber noch einmal dazu auf, die korrekten Disketten in beide Laufwerke einzulegen und dann erst <RETURN> zu drücken.

Besitzen Sie nur ein Diskettenlaufwerk (das eingebaute des Amiga 500), so wird die übliche Anwendungsform des Befehls für Sie wohl *diskcopy df0: TO df0:* sein. Genau wie beim Diskettenkopieren mit nur einem Laufwerk in der Workbench werden Sie nun immer wieder im Wechsel aufgefordert, die Original-Diskette oder die Diskette, die die Kopie aufnehmen soll, einzulegen. Eine gute Beschriftung beider Disketten ist für diesen Fall sehr wichtig und erleichtert Ihnen die Arbeit.

Egal, ob Sie ein oder zwei Diskettenlaufwerke besitzen, ist es in jedem Fall wichtig, Vorkehrungen zu treffen, damit nicht die Original-Diskette durch einen Bedienungsfehler zerstört wird. Der *diskcopy*-Befehl ist weitgehend gegen solche Bedienungsfehler geschützt, aber sie wissen ja: »Sicher ist sicher !« Deshalb schützen Sie bitte die Original-Diskette vor eventuellem Überschreiben, indem Sie das Schreibschutzplättchen in eine Position bringen, bei der ein kleines Loch entsteht. Die Diskette, die die Kopie aufnehmen soll, darf natürlich nicht schreibgeschützt sein.

Die auf einem dieser beiden Wege erzeugte Kopie wird für Sie nicht vom Original zu unterscheiden sein. (Das Betriebssystem kann die beiden Disketten allerdings als unterschiedlich erkennen. Die Zeit der letzten Schreiboperation wird nämlich auf der Kopie vermerkt und kann zur eindeutigen Identifikation einer Diskette verwendet werden.) Sie enthält alle Daten des Originals und trägt auch denselben Namen. Falls die Original-Diskette irgendwann einmal fehlerhaft wird, können Sie getrost die Kopie verwenden. Dies gilt leider nur in dem Fall, in dem der Hersteller der Programme, die sich eventuell auf dem Original befinden, diese nicht durch besondere Tricks vor dem Kopieren geschützt hat. Diese Unsitte

der Software-Hersteller ist bei anderen Computern schon weit verbreitet und wird Ihnen gewiß auch auf dem Amiga bald das Leben unnötig schwer machen.

Wenn Sie wollen, daß die Kopie einen anderen Namen als die Originaldiskette erhält, so können Sie diesen Namen gleich beim Kopieren verwenden. Dazu dient der Parameter <name> hinter dem Schlüsselwort NAME.

Beispiele:

```
1>diskcopy df1: to df0:
1>diskcopy from df0: df0:
1>diskcopy df0: to df1: NAME "Kopie der Diskette 1"
```

Der erste dieser Befehle kopiert die Diskette, die im ersten externen Laufwerk liegt, auf die Diskette im internen Laufwerk. Der zweite kopiert mit nur einem Laufwerk, wobei Sie zwischendurch immer wieder zum Wechsel zwischen Original-Diskette und Kopie aufgefordert werden. Und der letzte Befehl kopiert vom internen zum externen Laufwerk und gibt der Diskette im externen Laufwerk gleichzeitig den Namen *Kopie der Diskette 1*.

diskdoctor

Format: diskdoctor *DRIVE* <laufwerkname>

Zweck: dient zum Wiederherstellen defekter Disketten.

Erläuterung: Der Befehl *diskdoctor* kann unter gewissen Umständen Dateien beziehungsweise ganze Disketten retten, die durch Bedienungsfehler oder fehlerhafte Programme zerstört oder beschädigt wurden. Solche Fehler verkündet Ihnen das CLI mit der Fehlermeldung:

```
Disk structure corrupt. Use DiskDoctor to fix it!
```

In solchen Fällen sollten Sie zunächst (am besten mit *diskcopy*; siehe oben) eine Kopie der fehlerhaften Diskette anlegen. Legen Sie Ihre Workbench-Diskette in das interne Laufwerk und geben Sie dann den folgenden Befehl ein:

```
1>diskdoctor df0:
```

Dann erscheint die Meldung

```
Disk Doctor V1.3
Insert disk to be corrected and press RETURN
```

Nehmen Sie dann die Workbench-Diskette aus dem Laufwerk und schieben Sie die Kopie der defekten Diskette ein. Deren Schreibschutz darf dabei nicht aktiviert sein. Drücken Sie nun auf die [Return]-Taste. *Diskdoctor* versucht dann die Diskette zu »reparieren«, was geraume Zeit dauern kann. *Diskdoctor* gibt dabei einige Meldungen aus, die besagen, wie weit der Reparaturprozeß ist. Wenn das Programm dabei gelöschte Dateien oder Directories findet und wiederherstellt, erscheint immer die Meldung:

```
Replacing dir  <Directoryname>
```

beziehungsweise:

```
Replacing file <Dateiname>
```

Wenn die Reparatur beendet ist, erscheint die Meldung:

```
Now copy files required to a new disk and reformat this disk
```

Kopieren Sie daraufhin alle Dateien, die Sie von der defekten Diskette benötigen, auf eine andere, am besten frisch initialisierte (formatierte) Diskette. Dann können Sie versuchen, die defekte Diskette (also das Original) neu zu formatieren (mit *Initialize* auf der Workbench oder mit *format* im CLI.) Sollten dabei Fehler auftauchen oder gibt *diskdoctor* während seiner Arbeit die folgende Fehlermeldung aus, sollten Sie diese Diskette lieber in den Müll werfen oder zum Händler zurückbringen:

```
Hard Error Track <nn> Surface <n>
```

Der *diskdoctor*-Befehl kann nämlich echte (physikalische) Diskettenfehler, die bei der Produktion oder durch falsche Handhabung der Diskette entstanden sind, nicht beheben. Er repariert nur »Software-Fehler« und stellt Dateien wieder her, die Sie vielleicht versehentlich gelöscht haben.

echo

Format: echo <*Text*>

Zweck: dient zum Ausgeben eines Textes (einer Meldung) im CLI-Fenster.

Erläuterung: Der Text, der hinter echo (in Anführungsstrichen) als Parameter angegeben ist, wird als Folge des Befehls im CLI-Fenster ausgegeben. Dies ist im allgemeinen nur sinnvoll in Kommandofolgen und bei Befehlen, die im Hintergrund ausgeführt werden. Man kann den Text allerdings auch auf diese Weise zu einem Gerät schicken. Die Befehlsfolge echo >PRT: »Hallo Drucker !« gibt den Text »Hallo Drucker !« zum Beispiel auf einem eventuell angeschlossenen Drucker aus.

Beispiele:

```
1>echo "Hello world!"
1>echo >PRT: "Hello world!"
```

Der erste dieser beiden Befehle schreibt die Meldung »Hello world!« auf den Bildschirm, die zweite auf den Drucker (falls Sie einen besitzen, angeschlossen und eingeschaltet haben).

ed

Format: ed *FROM* <dateiname> *SIZE* <*n*>

Zweck: dient zum Anlegen und Modifizieren von Dateien, die druckbare Texte enthalten.

Erläuterung: Gibt man den Befehl *ed*, so gelangt man in einen einfachen »Editor«. Ich verwende das Wort »Editor« und nicht etwa »Textverarbeitungsprogramm«, um dieses einfache Programm von komfortableren Werkzeugen – wie zum Beispiel *TextCraft* – abzusetzen. Der Editor *ed* kennt keine verschiedenen Zeichensätze, kennt keine Möglichkeit zum Unterstreichen oder Fettdrucken oder für die seitenweise Bearbeitung von Texten. Er ist nur für die Bearbeitung von reinen Textdateien – zum Beispiel Programmen und Kommandofolgen – geeignet. Eine genaue Beschreibung von *ed* ist in einem folgenden Kapitel zu finden.

Gibt man als Parameter <dateiname> den Namen einer schon existierenden Datei an, so wird *ed* gestartet und diese Datei eingelesen, damit sofort mit ihrer Bearbeitung begonnen werden kann. Existiert diese Datei noch nicht, so wird eine neue Textdatei dieses Namens angelegt.

Normalerweise können mit *ed* Texte bis zu einer maximalen Größe von etwa 40 000 Anschlägen (circa 20 Schreibmaschinenseiten) bearbeitet werden. Dies reicht für die einfachen Anwendungen, für die *ed* gedacht ist, üblicherweise völlig aus. Benötigen Sie mehr Platz, können Sie hinter den Namen der zu bearbeitenden Datei noch das Schlüsselwort SIZE und die Anzahl der Buchstaben schreiben, mit denen *ed* höchstens »fertig werden« muß. Hier kann man natürlich auch kleinere Zahlen als 40 000 eingeben, wenn die zu bearbeitende Datei sehr klein ist und der Platz im Speicher des Amiga sehr eng ist.

Beispiele:

```
1>ed s/startup-sequence
1>ed Dissertation SIZE 100000
```

Der erste dieser Befehle startet den Texteditor *ed* und sorgt dafür, daß nach dem Start gleich die Datei *startup-sequence* bearbeitet werden kann. Der zweite Befehl lädt die Datei *Dissertation* und legt Platz für die Bearbeitung von 100 000 Zeichen an. (*ed* ohne jeden Parameter ergibt eine Fehlermeldung.)

edit

Format: edit *FROM* <dateiname1> TO <dateiname2>

WITH <editdatei> VER <ausgabe> OPT P <n> OPT W <n>

Zweck: dient zum Anlegen und Modifizieren von Dateien, die druckbare Texte enthalten.

Erläuterung: Der Befehl *edit* aktiviert einen sehr altmodischen, aber auch sehr leistungsfähigen Texteditor, mit dem Sie ähnliche Aufgaben erledigen können wie mit *ed.* Es handelt sich dabei um einen sogenannten »Zeileneditor«, wie sie früher auf wesentlich primitiveren Computern weit verbreitet waren. Zeileneditoren sind unhandlich zu bedienen und erlauben immer nur die Bearbeitung einer aktuellen Zeile. Der wesentliche Vorteil von *edit* gegenüber *ed* liegt allerdings darin, daß *ed* über die Tastatur bedient werden muß und die Bearbeitung eines Textes deshalb nicht mit einer Kommandofolge automatisiert werden kann. Wollen Sie in zehn langen Texten sämtliche Vorkommnisse zweier Wörter durch zwei andere Wörter ersetzen, müssen Sie *ed* aufrufen und dann zehnmal nacheinander eine Datei einlesen, das erste Wort suchen und ersetzen lassen, das zweite Wort suchen und ersetzen lassen und die Datei wieder abspeichern. *Edit* kann die Befehle, mit denen ein Text editiert werden soll, auch von einer Textdatei empfangen. Sie können *edit* auf diese Weise automatisch (ohne Aufsicht) langwierige Operationen mit einer oder mehreren großen Dateien durchführen lassen.

Sie werden *edit* im Normalfall nie benötigen und die anderen Editoren *ed* und *emacs* auch als wesentlich angenehmer empfinden. Benötigen Sie jedoch die Möglichkeiten von *edit*, so finden Sie eine ausführliche Beschreibung im *Amiga-DOS Handbuch* (erschienen im Markt&Technik-Verlag, Nr. 90465, Haar b. München, 1987).

else

Format: else

Zweck: dient zur bedingten Ausführung von Befehlen in Kommandofolgen.

Erläuterung: Das Kommando *else* steht üblicherweise für sich allein in einer Zeile einer Kommandofolge. (Es ist nicht sinnvoll, *else* direkt im CLI zu verwenden.) Vor *else* muß in derselben Kommandofolge noch ein *if*-Befehl auftauchen. Nach *else* muß in derselben Kommandofolge noch ein *endif*-Befehl folgen. Hinter einem *if*-Befehl steht immer eine Bedingung. (Vergleichen Sie die Beschreibung des *if*-Befehls.) Trifft diese Bedingung zu, werden die Kommandos zwischen *if* und *else* ausgeführt und die Kommandos zwischen *else* und *endif* ignoriert (übersprungen). Trifft die Bedingung nicht zu, werden die Kommandos zwischen *if* und *else* übersprungen und die Kommandos zwischen *else* und *endif* ausgeführt. Solche *If-else-endif*-Gruppierungen können auch geschachtelt auftauchen. Das heißt, zwischen *else* und *endif* könnte zum Beispiel eine weitere *If-else-endif*-Gruppierung stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« *if*-Befehls nicht zutrifft.

Mehr zu den Befehlen *if*, *else* und *endif* können Sie im übernächsten Kapitel *Einfacher Arbeiten mit Kommandofolgen* erfahren.

endcli

Format: endcli

Zweck: dient zum Beenden eines aktiven CLI-Task.

Erläuterung: Der Befehl *endcli* beendet den CLI-Task, in dem der Befehl erteilt wird. Das entsprechende Fenster wird geschlossen, nachdem eine entsprechende Meldung ausgegeben wird. (Diese Meldung wird in der Startup-sequence mit >NIL: ins »Nichts« geschickt und ist deshalb nicht zu sehen.)

Mit diesem Befehl sollte man sehr vorsichtig umgehen! Er sollte nur erteilt werden, wenn »im Hintergrund« hinter dem CLI-Fenster noch die Workbench sichtbar ist oder noch ein oder mehrere andere CLI-Fenster offen sind. Ist die Workbench nicht geladen worden und das geschlossene Fenster war das einzige CLI-Fenster, »hängt« der Amiga – er nimmt dann keinerlei Befehle über die Tastatur oder die Maus mehr an.

endif

Format: endif

Zweck: dient zur bedingten Ausführung von Befehlen in Kommandofolgen.

Erläuterung: Das Kommando *endif* steht meistens für sich allein in einer Zeile einer Kommandofolge. (Es ist nicht sinnvoll, *endif* direkt im CLI zu verwenden.) Vor *endif* muß in derselben Kommandofolge noch ein *if*-Befehl auftauchen. Zwischen diesem *if* und *endif* kann in derselben Kommandofolge ein *else*-Befehl stehen. Hinter einem *if*-Befehl folgt immer eine Bedingung. (Vergleichen Sie die Beschreibung des *if*-Befehls.) Trifft diese Bedingung zu, werden die Kommandos zwischen *if* und *else* ausgeführt und die Kommandos zwischen *else* und *endif* ignoriert (übersprungen). Trifft die Bedingung nicht zu, werden die Kommandos zwischen *if* und *else* übersprungen und die Kommandos zwischen *else* und *endif* ausgeführt. Solche *If-else-endif*-Gruppierungen können auch geschachtelt auftauchen. Das heißt, zwischen *else* und *endif* könnte zum Beispiel eine weitere *If-else-endif*-Gruppierung stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« *if*-Befehls nicht zutrifft.

Mehr zu den Befehlen *if*, *else* und *endif* können Sie im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen* erfahren.

execute

Format: execute <kommandofolge> *<argumentl1>* *<argumentl2>* ...

Zweck: dient zum Ausführen einer Liste von CLI-Kommandos, die in einer Textdatei gespeichert sind.

Erläuterung: Der Befehl *execute* dient zur »Automatisierung« des CLI. Diese Möglichkeit, längere Aufgabenfolgen ohne manuelles Zutun ablaufen zu lassen, ist einer der wesentlichen Vorteile des CLI gegenüber der Workbench. Hierzu werden die CLI-Befehle, die nacheinander ausgeführt werden sollen, in einer Textdatei gespeichert. Die Ausführung aller dieser Befehle kann mit execute <kommandofolge> »angestoßen« werden. Dabei ist <kommandofolge> der Name der Textdatei, die die CLI-Befehle enthält. In der Kommandofolge können die Argumente (kurze Texte), die dem *execute*-Befehl übergeben werden, als Parameter und auch für andere Zwecke verwendet werden.

Die Möglichkeiten von *execute* werden ausführlich im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen* erläutert.

failat

Format: failat <*n*>

Zweck: dient zum Ändern des Abbruch-Levels, welches erreicht werden muß, bis eine Kommandofolge abgebrochen wird.

Erläuterung: Der Befehl *execute* (siehe oben) führt eine Folge von CLI-Befehlen aus, die in einer Textdatei gespeichert sind. Tritt bei der Ausführung eines dieser Befehle ein Fehler auf, so bricht je nach Schwere des Fehlers eventuell die Ausführung der gesamten Folge sofort ab. Die restlichen Befehle werden nicht mehr aufgerufen. Ein sofortiger unbedingter Abbruch ist nicht immer wünschenswert. Manchmal möchte man, daß nur bei sehr schwerwiegenden Fehlern abgebrochen wird oder man möchte den Fehler selbst erkennen und innerhalb der Kommandofolge darauf reagieren. In diesem Fall muß man den Level, ab dem Kommandofolgen bei einem Fehler abgebrochen werden, heraufsetzen.

Hierzu dient der Befehl *failat*: Durch *failat* wird der Abbruch-Level auf <n> gesetzt oder der aktuelle Abbruch-Level ausgegeben, wenn Sie <n> nicht angeben. Der Befehl *failat 20* setzt den Abbruch-Level zum Beispiel auf 20 fest, so daß nur noch relativ schwere Fehler die aktuelle Kommandofolge unterbrechen können. Auf Wunsch können sogar noch höhere Werte verwendet werden. Das Amiga-Betriebssystem unterscheidet grob zwischen drei Levels:

5 für Warnungen (außergewöhnliche Situationen, die eigentlich kein Fehler sind)

10 für normale Fehler

20 und größer für besonders schwerwiegende Fehler, die nicht behebbar sind

Der »normale« Abbruch-Level ist 10. Mehr dazu aber im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen.*

fault

Format: fault *<n1> <n2> ... <n10>*

Zweck: dient zur Ausgabe einer (englischen) Fehlermeldung im Volltext zu einem bestimmten Fehlercode.

Erläuterung: Amiga-DOS meldet manchmal einen Fehler nur als Nummer. Wenn diese Nummer in einem Requester oder als Ausgabe eines fehlgeschlagenen CLI-Kommandos auftaucht, können Sie mit *fault* eine knappe (englische) Erläuterung dazu erhalten. Dazu brauchen Sie nur eine Liste der Fehlernummern (maximal 10) hinter *fault* anzugeben. Weitergehende Beschreibungen dieser Fehler, mögliche Fehlerursachen und Behebungsvorschläge finden Sie im Anhang dieses Buches.

Beispiele:

```
1>fault 103 203
fault 103: insufficient free store
fault 203: object already exists
```

filenote

Format: filenote *FILE* <dateiname> *COMMENT* <Kommentar>

Zweck: dient dazu, eine Datei mit einem Kommantar zu versehen und diesen wieder zu ändern.

Erläuterung: Das Betriebssystem des Amiga kann zu jeder Datei einen kurzen Kommentar von maximal 80 Anschlägen verwalten. Dieser Kommentar kann von Ihnen völlig frei gewählt werden. Er wird im allgemeinen zusätzliche Informationen über den Inhalt der Datei enthalten, die über das hinausgehen, was man in einem Dateinamen unterbringen kann. So könnte man zum Beispiel vermerken, daß es sich bei einer bestimmten Datei um die 3. Version handelt und ähnliches. Der Kommentar zu einer Datei kann auf Wunsch mit dem CLI-Befehl *list* (siehe unten) betrachtet, jedoch nicht geändert werden, und mit dem Befehl *filenote* geändert, aber nicht betrachtet werden. In der Workbench kann derselbe Kommentar mit dem Menübefehl *Info* betrachtet und geändert werden.

Um eine Datei mit einem Kommentar zu versehen, brauchen Sie nur den Namen dieser Datei hinter *filenote* anzugeben und hinter dem Schlüsselwort COMMENT dann den Text des Kommentars. Handelt es sich bei diesem Kommentar um mehrere Wörter, die durch Zwischenräume und/oder Sonderzeichen getrennt sind, muß der Kommentar in Anführungszeichen gesetzt werden.

Der Befehl *filenote* hat übrigens einige merkwürdige Eigenschaften. Eine neue Datei hat zunächst keinen Kommentar. Wird aber – zum Beispiel beim Kopieren – eine alte Datei mit einem neuen Inhalt überschrieben, so bleibt der alte Kommentar dieser Datei erhalten (der ja jetzt nicht mehr zutrifft). Es wird also nicht der Kommentar der Datei verwendet, die kopiert wurde, sondern der Kommentar der Datei, die durch den Kopiervorgang zerstört wurde.

Beispiele:

```
1>filenote FILE Brief2 COMMENT "Brief an Tante Elli"
1>filenote geheim "Das ist geheim!!!"
```

format

Format: format DRIVE <laufwerkname> NAME <diskettenname> *NOICONS*

Zweck: dient zum Formatieren und Initialisieren einer unbenutzten Diskette beziehungsweise Festplatte oder zum Löschen einer in Gebrauch befindlichen Diskette beziehungsweise Festplatte.

Erläuterung: Bevor eine Diskette oder eine Festplatte verwendet werden kann (bevor Directories und Dateien darauf angelegt werden können), muß sie formatiert werden. Bei diesem Vorgang wird der amorphen Struktur des Speichermediums, das ja nur eine Fläche magnetisierbaren Materials ist, ein Format »aufgezwungen«. Die Diskette besteht danach aus einer Folge von konzentrischen Spuren (engl. *tracks*) und jede dieser Spuren besteht aus einer bestimmten Anzahl von Blöcken oder Sektoren (engl. *sectors*). Nach dieser rein magnetischen Formatierung muß auch noch das zunächst leere Directory angelegt und initialisiert werden, das die Wurzel der Datei-Hierarchie auf dieser Diskette oder Festplatte bildet.

Diese beiden Aufgaben erfüllt der CLI-Befehl *format.* Seine Anwendung ist unmißverständlich. Er benötigt zwei Parameter, die beide von einem Schlüsselwort eingeleitet werden. Der erste <laufwerkname> gibt an, in welchem Diskettenlaufwerk sich die zu formatierende Diskette befindet (im allgemeinen wohl df0: oder df1:), der zweite <diskettenname> gibt den Namen an, den die frisch formatierte Diskette danach tragen soll. Enthält dieser Name Leerstellen, muß er in Anführungszeichen gesetzt werden.

Format ist nur dann nötig, wenn Sie eine leere Diskette benötigen, um auf ihr einzelne neue Dateien anzulegen. Wollen Sie eine Diskette kopieren (mit *diskcopy*), so brauchen Sie die neue Diskette, die die Kopie aufnehmen soll, nicht zu formatieren. Das erledigt *diskcopy* gleich mit!

Wenn Sie die neuformatierte Diskette nur im CLI verwenden wollen, sollten Sie das Schlüsselwort NOICONS am Ende des *format*-Befehls verwenden. Dann erscheint kein Mülleimer auf dieser Diskette und Sie gewinnen ein paar Bytes zusätzlichen Speicherplatz.

Beispiele:

```
1>format DRIVE df0: NAME Disk1
1>Format DRIVE df1: NAME "Disk 1" NOICONS
```

Der erste Befehl formatiert die Diskette im internen Laufwerk und gibt ihr den Namen *Disk1*. Im Diskettenfenster dieser Diskette erscheint später auf der Workbench ein Mülleimer. Der zweite Befehl formatiert die Diskette im ersten externen Laufwerk und gibt ihr den Namen *Disk 1*. Diese Diskette enthält danach keine Piktogramme.

if

Format: if <bedingung>

Zweck: dient zum bedingten Ausführen oder zum Überspringen einer Teilfolge von CLI-Befehlen innerhalb einer Kommandofolge.

Erläuterung: Den Befehl *if* werden Sie nie direkt dem CLI erteilen. Er ist nur in Kommandofolgen sinnvoll. If funktioniert ähnlich wie die IF-Anweisung in einer Programmiersprache (zum Beispiel BASIC). Hinter *if* folgt immer eine Bedingung, die das CLI überprüft, wenn es auf diesen Befehl trifft. Trifft diese Bedingung nicht zu, ignoriert das CLI alle Befehle, bis es auf einen *else*-Befehl oder einen *endif*-Befehl trifft. Trifft die Bedingung jedoch zu, so werden die folgenden Befehle in der Kommandofolge bearbeitet, bis die Befehle *else* oder *endif* auftreten. Wird dann der Befehl *else* zuerst gefunden, werden alle folgenden Befehle bis zu *endif* ignoriert. Solche *If-else-endif*-Gruppierungen können auch geschachtelt auftauchen. Das heißt, zwischen *else* und *endif* könnte zum Beispiel eine weitere *If-else-endif*-Gruppierung stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« *If*-Befehls nicht zutrifft.

Die folgenden Bedingungen können zusammen mit *if* verwendet werden:

WARN trifft dann zu, wenn der letzte Befehl einen Fehlercode ergeben hat, der größer oder gleich 5 ist.

ERROR trifft dann zu, wenn der letzte Befehl einen Fehlercode ergeben hat, der größer oder gleich 10 ist.

FAIL trifft dann zu, wenn der letzte Befehl einen Fehlercode ergeben hat, der größer oder gleich 20 ist.

Mit diesen Bedingungen können Kommandofolgen vom Erfolg vorangehender Befehle abhängig gemacht werden. Damit solche Bedingungen sinnvoll sind, muß unbedingt der Fehler-Level mit *failat* hochgesetzt werden. Ist er zum Beispiel kleiner als 10, so wird das CLI nie bis zu einer Bedingung *FAIL* kommen, weil die Ausführung der Kommandofolge schon vorher abgebrochen wird.

EXISTS <name> trifft dann zu, wenn eine Datei oder ein Directory mit dem Namen <name> existiert (zum Beispiel *IF EXISTS Brief1*).

<text1> EQ <text2> trifft dann zu, wenn <text1> gleich <text2> ist (wobei Groß- beziehungsweise Kleinschreibung keine Rolle spielen. Entweder <text1> oder <text2> ist dabei üblicherweise ein Parameter der Kommandofolge.

Setzt man schließlich das Schlüsselwort NOT vor eine der bislang beschriebenen Bedingungen (zum Beispiel *IF NOT EXISTS Brief1*), so wird deren Ergebnis umgekehrt.

Mehr zum *if*-Befehl aber im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen.*

info

Format: info

Zweck: dient zur Information über den Zustand des gesamten Dateisystems.

Erläuterung: Der Befehl *info* gibt eine Liste aus, in der für jede dem Betriebssystem im Moment bekannte Diskette, Festplatte und so weiter die folgenden Informationen ausgegeben werden:

- das Laufwerk, in dem sich die Diskette/Festplatte befindet
- die Größe der Diskette (in Kbyte)
- wieviel Kbyte darauf belegt sind
- wieviel Kbyte darauf frei sind
- welcher Prozentsatz belegt ist
- wieviele Diskettenfehler darauf registriert wurden
- ob der Schreibschutz aktiviert ist (*Read Only*) oder nicht (*Read/Write*)
- den Namen der Diskette/Festplatte

install

Format: install *DRIVE* <laufwerkname>

Zweck: macht eine formatierte Diskette zur startfähigen Diskette.

Erläuterung: Der Befehl *install* ist nur für fortgeschrittene Anwender gedacht. Mit ihm werden einige der Informationen auf die Diskette <laufwerkname> (zum Beispiel *df0:* oder *df1:*) geschrieben, die der Amiga beim Start oder zum Neustart benötigt. Eine solche Diskette wird vom Amiga 500 dann akzeptiert, wenn er beim Start eine »Workbench-Diskette« verlangt. Falls nicht noch weitere Vorkehrungen getroffen werden, ist die so entstehende Diskette aber reichlich unbrauchbar. Zu diesen Vorkehrungen gehört das Einrichten einer Reihe von Directories auf dieser Diskette (zum Beispiel *devs*, *fonts* und so weiter) sowie das Füllen dieser Directories mit Dateien. Bevor Sie den Befehl anwenden, sollten Sie mehr über die Dateien wissen, die Amiga-DOS unbedingt benötigt, als Ihnen dieses Buch vermitteln kann. Vom (vorsichtigen) Experimentieren kann und soll durch diese warnenden Worte aber niemand abgehalten werden.

join

Format: join <name1> <name2> ... <*name15*> AS <neuername>

Zweck: verbindet bis zu 15 alte Dateien zu einer neuen Datei, indem die Daten der 15 alten Dateien hintereinandergehängt werden.

Erläuterung: Der Befehl *join* (engl. *to join* = für *verbinden, verknüpfen*) dient zur Verknüpfung von Dateien. Dies kann aus den verschiedensten Gründen heraus wichtig sein. Eine solche Verknüpfung funktioniert aber meist nur bei Dateien, die aus einer Folge gleichförmiger Daten bestehen. Textdateien können zum Beispiel ohne weiteres verknüpft werden. Mit *join* kann man aus mehreren kurzen Textdateien eine lange machen. Bei Dateien mit komplexem Format, wie zum Beispiel Bildern, ist ein *join* meist nicht möglich.

Beachten Sie bitte, daß die neue Datei nicht denselben Namen tragen kann wie eine der darin zu verschmelzenden Dateien!

Beispiel:

```
1>join Brief1 Brief2 "Brief 3" AS Briefe
```

Dieser Befehl erzeugt eine neue Datei namens Briefe im aktuellen Directory, in der hintereinander alle Daten aus den Dateien *Brief1*, *Brief2* und *Brief 3* stehen. Die Dateien *Brief1*, *Brief2* und *Brief 3* bleiben dabei erhalten.

lab

Format: lab <name>

Zweck: setzt eine Sprungmarke in einer Kommandofolge.

Erläuterung: Der Befehl *lab* markiert die Zeile einer Kommandofolge, an der er steht, als das mögliche Ziel eines entsprechenden *skip*-Kommandos. Stößt das CLI in einer Kommandofolge auf den Befehl *skip <name1>*, so werden alle Befehle übersprungen, bis eine Zeile mit dem Befehl *lab <name2>* am Anfang gefunden wird und <name1> und <name2> gleich sind.

Mehr zum *lab*-Befehl aber im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen.*

list

Format: list *DIR* <name>

PAT <muster> P <muster>

KEYS DATES NODATES

TO <datei> S <suchtext>

SINCE <datum1> UPTO <datum2>

QUICK

Zweck: gibt eine Menge von Informationen über eine Datei oder eine Gruppe von Dateien aus.

Erläuterung: Der Befehl *list* dürfte der CLI-Befehl mit den meisten möglichen Parametern und Schlüsselwörtern sein. Er arbeitet grundsätzlich ähnlich wie der *dir*-Befehl, kann aber nicht nur Namen, sondern auch sehr ausführliche Informationen über eine Datei anzeigen. Dazu gehören Name und Größe (in Bytes) der Datei, eventuelle Schreib-/Lese-Schutzflags, das Datum und die Uhrzeit der letzten Modifikation und schließlich der Kommentar (siehe auch den CLI-Befehl *filenote*). Verschiedene Optionen steuern, welche Informationen ausgegeben und für welche Dateien sie ausgegeben werden.

Gibt man *list* ohne den ersten Parameter <name> an, so werden Informationen über alle Dateien im aktuellen Directory aufgelistet (sofern nicht andere Optionen diese Menge einschränken). Ist <name> ein Dateiname, so werden nur Informationen über diese Datei ausgegeben. Ist <name> der Name eines Directory, so werden Informationen über alle Dateien dieses Directory aufgelistet (sofern nicht andere Optionen diese Menge einschränken).

Gibt man das Schlüsselwort KEYS mit an, so erhält man eine technische Information zu jeder Datei, mit der man defekte Dateien eventuell wiederherstellen kann. Das Schlüsselwort NODATES unterdrückt die Ausgabe von Datums- oder Uhrzeit-Informationen zu den Dateien. Das Schlüsselwort QUICK bewirkt, daß nur noch die Namen der Dateien und Directories ausgegeben werden und reduziert deshalb *list* weitgehend auf die Funktion, die auch der *dir*-Befehl erfüllt.

Mit der Option *TO <datei>* können Sie die Liste, die *list* normalerweise auf den Bildschirm ausgeben würde, in eine Datei schicken, um sie dort zu betrachten oder zu bearbeiten.

Mit den Schlüsselwörtern P, S, SINCE und UPTO kann die Menge der Dateien aus einem Directory, über die Informationen ausgegeben werden, eingeschränkt werden. Gibt man *P <muster>* oder *PAT <muster>* zusammen mit *list* ein, so werden nur Dateien berücksichtigt, die in dieses Muster passen. Gibt man *S <suchtext>* an, so werden nur Dateien berücksichtigt, die in ihrem Namen den bewußten Text enthalten. (S ist also eine vereinfachte Fassung von PAT.) *SINCE <datum1>* bewirkt, daß nur Dateien aufgelistet werden, die seit <datum> (engl. *seit = since*) noch modifiziert wurden. *UPTO <datum2>* schränkt die Ausgabe auf Dateien ein, die nach <datum2> nicht mehr modifiziert wurden.

Der Befehl *list* kennt so viele Optionen, daß Sie am besten ein wenig damit experimentieren sollten. Vergessen Sie auch nicht, daß Sie die Ausgabe, also die Informationen über die Dateien, mit TO in eine Datei schicken und sie dort in Ruhe betrachten können (zum Beispiel auch mit dem *Notepad*).

Beispiele:

```
1>list SINCE 01-Jan-87 TO DirListe QUICK
1>list :C S ed
```

Der erste Befehl erzeugt eine Datei namens *Dirliste* im aktuellen Directory, die eine Liste aller Dateinamen enthält, die nach dem 1. Januar 1987 noch geändert wurden. Außer den Namen werden keine Informationen abgespeichert. Der zweite Befehl listet Namen und einige weitere Informationen über alle Dateien im Directory c: auf, die in ihrem Namen den Suchtext »ed« enthalten.

loadwb

Format: loadwb

Zweck: Startet die Workbench, falls sie noch nicht aktiv ist.

Erläuterung: Unter normalen Umständen wird beim Starten des Amiga die Workbench aktiv und bleibt immer aktiv, auch während andere Programme gestartet werden und laufen. Das Starten der Workbench geschieht in der Startup-Sequence bei jedem Neustart automatisch. Man kann diesen Start der Workbench jedoch verhindern und beim Start gleich in das CLI gelangen. Möchte man dann trotzdem wieder in der Workbench arbeiten, müssen Sie dazu den Befehl *loadwb* einmal aufrufen.

Mehr dazu aber im Kapitel *Tips zum CLI*.

makedir

Format: makedir <dir>

Zweck: dient zum Anlegen eines neuen Directory.

Erläuterung: Der Befehl *makedir* ist immer dann nötig, wenn Sie Dateien in ein neues Directory legen wollen. Vorher muß dieses neue Directory explizit mit *makedir* angelegt werden. Dies ist eine Sache, die Sie nicht oft genug machen können, um Ihre Dateien gut gegliedert zu halten.

Directories können genau wie Dateien mit dem Befehl *delete* gelöscht werden. Sie müssen dazu aber vollständig leer sein!

Beispiele:

```
1>makedir NeuDir
1>makedir :c/selten
```

Der erste Befehl legt ein neues Directory namens *NeuDir* an, das im aktuellen Directory liegt. Der zweite Befehl legt ein neues Directory namens *selten* im Directory *c* auf der aktuellen Diskette an (das Directory *c* muß dazu schon existieren).

microemacs

Format: microemacs <*name*>

Zweck: dient zum Anlegen und Modifizieren von Dateien, die druckbare Texte enthalten.

Erläuterung: Microemacs ist der dritte und komfortabelste Editor des Amiga 500. Er liegt unter Umständen nicht mehr jedem Amiga 500 bei, wenn Sie dieses Handbuch lesen. Sie können ihn dann jedoch zusammen mit anderen Hilfsprogrammen auf der Diskette *Amiga-Toolkit* bei Ihrem Amiga-Händler kaufen. Mehr über die Bedienung dieses sehr komfortablen Text-Editors erfahren Sie in einem folgenden Kapitel.

Wenn Sie *microemacs* ohne Parameter aufrufen, wird das Programm gestartet und erscheint auf dem Bildschirm. Sie können dann Texte einlesen, ändern und wieder abspeichern. Wenn Sie den Parameter <name> angeben, wird diese Datei gleich nach dem Start geladen und Ihnen am Bildschirm zur Bearbeitung präsentiert.

mount

Format: mount <gerätename>

Zweck: dient zum »Anmelden« eines Gerätes beim Amiga-DOS, das durch einen zusätzlichen Treiber angesprochen wird, der nicht zur »Standardausstattung« der Workbench-Diskette gehört.

Erläuterung: Wenn Sie zusätzliche Geräte an Ihren Amiga 500 anschließen, müssen Sie dies Amiga-DOS »mitteilen«. Hierzu gehört einmal der *binddrivers*-Befehl (siehe oben), der den »Treiber« dieses Geräts in der Expansion-Schublade sucht, ihn lädt und aktiviert. Der zweite Schritt ist die Anmeldung dieses Geräts mit einem Namen, unter dem es dann zum Beispiel in CLI-Befehlen angesprochen werden kann; hierzu dient *mount*. Nach *binddrivers* sollte deshalb *mount* mit dem Gerätenamen als Parameter aufgerufen werden. *Mount* benötigt allerdings weitere Angaben, um zu wissen, welches Gerät mit diesem Namen angesprochen werden soll, wie sein Treiber heißt, und wie sich das Gerät verhält. Diese Angaben entnimmt es der Datei *mountlist* im logischen Laufwerk *DEVS:*.

Der Inhalt dieser Datei ist nur für sehr fortgeschrittene Benutzer verständlich. Sie benötigen dazu Informationen, die ich Ihnen im Rahmen dieses Buches nicht vermitteln kann. Der Hersteller eines Gerätes, das mittels *mount* angemeldet werden muß, wird in der Regel aber eine korrekte *mountlist* zusammen mit dem Gerätetreiber auf einer separaten Diskette mitliefern. Sie brauchen diese dann nur in das Directory *devs* der Workbenchdiskette zu legen, mit der Sie starten und brauchen sich um den Inhalt der *mountlist* nicht zu kümmern. Wenn Sie mehrere Geräte auf diese Weise anschließen wollen, müssen Sie die verschiedenen, von den Geräteherstellern gelieferten *mountlist*-Dateien mit dem Befehl *join* verketten und dann unter dem Namen *mountlist* in das logische Laufwerk *DEVS:* legen.

Mount steht – wie *binddrivers* – meist in der Startup-Sequence und wird nicht direkt im CLI-Fenster eingegeben.

newcli

Format: newcli *<fenster>*

Zweck: dient zum Erzeugen eines neuen CLI-Task in einem neuen Fenster.

Erläuterung: Der Befehl *newcli* startet – sofern noch genügend Speicherplatz frei ist – einen neuen Task, in dem auch das CLI läuft. Dieser Task erhält ein eigenes Fenster, in dem sich die Kommunikation mit dem Benutzer abspielt.

Geben Sie *newcli* ohne Parameter ein, erhalten Sie ein Standardfenster für die Kommunikation mit dem neuen CLI-Task. Dieses Fenster kann danach beliebig verschoben, vergrößert und verkleinert werden. Möchten Sie aber sofort ein Fenster passender Größe und Position haben, so können Sie dies dem Befehl durch den Parameter <fenster> mitteilen.

<Fenster> ist ein Text der Form CON:<x>/<y>/<breite>/<höhe>/<titel>. Er enthält im wesentlichen (durch Schrägstriche getrennt) den Abstand vom oberen Bildschirmrand, vom unteren Bildschirmrand, die Breite und die Höhe des Fensters und dessen Titel. Enthält der Titel Leerzeichen oder Sonderzeichen, muß der ganze Parameter <fenster> in Anführungszeichen gesetzt werden. *CON:* teilt dem Betriebssystem des Amiga mit, daß sich das Gerät *CON:* (Tastatur und Bildschirm) um dieses Fenster kümmern soll. Ein anderes Gerät kann sich sinnvollerweise aber kaum um ein CLI-Fenster kümmern und sollte hier deshalb nicht angegeben werden. Fortgeschrittene Benutzer können es aber vielleicht auch mit *SER:* (der seriellen Schnittstelle, an der dazu ein anderer Computer angeschlossen sein muß) oder *RAW:* versuchen.

Beispiele:

```
1>newcli "CON:10/30/600/100/Neues CLI"
1>newcli "RAW:10/30/600/100/RAW-CLI"
```

path

Format: path *<name1> <name2> ... <name10> ADD RESET SHOW*

Zweck: dient zur Anzeige und zum Ändern des Suchpfades, dessen Directories Amiga-DOS nacheinander durchsucht, um ein Programm zu starten, von dem nicht der vollständige Pfadname bekannt ist.

Erläuterung: Die geschickte Verwendung des Befehls *path* ist sehr wichtig, damit Sie im CLI problemlosen Zugriff auf möglichst viele Programme in verschiedenen Directories haben. *Path* verwaltet eine Liste von Directories. Das erste Directory in dieser Liste ist immer das aktuelle Directory, das letzte ist das logische Laufwerk *C:*. Dazwischen können beliebige andere Directories liegen. Welche Directories der Pfad gerade enthält, können Sie sich anzeigen lassen, indem Sie *path* ohne alle Parameter oder nur mit dem Schlüsselwort SHOW (engl. *to show = zeige*) verwenden. Wenn Sie SHOW verwenden, überprüft *path* nicht, ob alle Directories im Suchpfad noch erreichbar sind. Wenn Sie hingegen *path* ohne Parameter eingeben und sich im Suchpfad ein Directory auf einer Diskette befindet, die gerade nicht eingelegt ist, wird diese Diskette mit einem Requester angefordert. Wenn Sie also wirklich nur schnell sehen wollen, wie der Pfad aussieht, verwenden Sie *path SHOW*.

Die beiden anderen Schlüsselwörter dienen zur Veränderung des Suchpfads. Geben Sie eine Liste von Directorynamen hinter *path* an und beenden den Befehl mit ADD (engl. *to add = hinzufügen*), werden diese Directories in der genannten Reihenfolge vor *C:* in den Suchpfad eingehängt. Wenn Sie hingegen das Schlüsselwort RESET (engl. *to reset = zurücksetzen*), wird der Suchpfad in seinen »Urzustand« zurückversetzt. Er enthält dann nur noch das aktuelle Directory und das logische Laufwerk *C:*.

Beispiele:

```
1>path Briefe :Romane ADD
1>path RESET
1>path
1>path SHOW
```

Der erste Befehl verlängert den Suchpfad um das Directory *Briefe* im aktuellen Directory und um das Directory *Romane* im Wurzeldirectory der aktuellen Diskette. Der zweite Befehl löscht alle Directories außer *C:* und dem aktuellen Directory aus dem Suchpfad. Der Befehl *path* allein listet den aktuellen Suchpfad auf und versucht, auf jedes der Directories zuzugreifen. Der letzte Befehl hingegen listet den Suchpfad nur auf.

prompt

Format: prompt <*text*>

Zweck: dient zum Ändern des CLI-Prompts, also der Meldung, mit der das CLI dem Benutzer mitteilt, daß es bereit ist, den nächsten Befehl entgegenzunehmen.

Erläuterung: »Normalerweise« ist das Prompt, mit dem das CLI dem Benutzer mitteilt, daß es bereit ist, den nächsten Befehl entgegenzunehmen, eine Zahl und das Größer-Zeichen >. Die Zahl dient dazu, verschiedene CLI-Fenster zu unterscheiden, sofern man mehrere davon hat. Sie ist gleich der laufenden Nummer des CLI-Tasks, weswegen man vom allerersten CLI immer mit *1>* begrüßt wird. Der Befehl *prompt* ändert diese Meldung. Statt 1> erscheint <text>. Wie üblich, müssen Sie <text> in Anführungsstriche setzen, wenn er Leerstellen enthält. Wenn Sie innerhalb von <text> den Platzhalter »%N« verwenden, so wird dieser später gegen die Task-Nummer des jeweiligen CLI-Fensters ersetzt.

Beispiele:

```
1>prompt "Deine Befehle Herr:"
Deine Befehle Herr: prompt "Was is ?"
Was is?prompt "Dies ist Task %N> "
Dies ist Task 1> prompt "%N>"
1>
```

protect

Format: protect *FILE* <dateiname> *FLAGS* <schalter>

Zweck: dient zum Ändern der Sicherungsschalter einer Datei.

Erläuterung: Das Betriebssystem des Amiga, Amiga-DOS, verwaltet zu jeder Datei 4 Sicherungsschalter, die einen Mißbrauch der Datei möglichst schwierig machen. Jeder dieser Schalter kann AN oder AUS sein und erlaubt oder verhindert bestimmte Operationen mit der Datei. Die vier Schalter (engl. *flags*) haben einbuchstabige Namen. Sie lauten:

r: die Datei kann gelesen werden (r für *read = lesen*)

w: die Datei kann geändert werden (w für *write = schreiben*)

d: die Datei kann gelöscht werden (d für *delete = löschen*)

e : die Datei kann als Programm ausgeführt werden (e für *execute = ausführen*)

Alle vier Schalter sind bei einer neuen Datei zunächst einmal gesetzt. Jede Datei kann also zu Beginn ihrer Existenz gelesen, geändert, gelöscht und ausgeführt werden. Mit *protect* (engl. *to protect = schützen*) werden nun alle Schalter auf AN gesetzt, deren Namen hinter dem Schlüsselwort FLAGS auftauchen. Der Befehl *protect* kann nur auf einzelne Dateien und (leider) nicht auf Gruppen von Dateien angewendet werden. Es werden immer nur die Schalter der Datei geändert, deren Name direkt hinter *protect* auftaucht.

Beispiel:

```
1>protect text1 FLAGS rw
```

Dieser Befehl erklärt die Datei *text1* als lesbar und änderbar. Sie kann aber nicht gelöscht oder als Programm behandelt werden.

quit

Format: quit *RC <Rückkehrcode>*

Zweck: dient zum Beenden einer Kommandofolge und der eventuellen Rückmeldung eines aufgetretenen Fehlers.

Erläuterung: Den Befehl *quit* werden Sie nie direkt an das CLI erteilen. Er ist nur in Kommandofolgen sinnvoll. Trifft das CLI in einer Kommandofolge auf *quit*, dann wird diese Kommandofolge sofort beendet. Kommandofolgen werden »ordnungsgemäß« mit ihrer letzten Zeile beendet, deshalb ist *quit* als eine Art »Notausgang« zu bezeichnen. Aus demselben Grund kann *quit* auch einen Rückkehrcode zurückgeben. Dieser spiegelt die Schwere des eingetretenen Fehlers wieder und kann in einem IF-Befehl abgefragt werden. Ist der Returncode größer als Null, kann dies zum Beispiel dazu führen, daß eine andere Kommandofolge abgebrochen wird.

Wenn in einer Kommandofolge zum Beispiel der Befehl *quit 20* ausgeführt wird, so trifft unmittelbar danach in anderen Kommandofolgen die Bedingung *if ERROR* oder *if FAIL* zu. Wird hingegen nur *quit* ausgeführt, treffen unmittelbar danach weder *if WARN*, noch *if ERROR* oder *if FAIL* zu.

Mehr zum *quit*-Befehl aber im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen.*

relabel

Format: relabel *DRIVE* <name> *NAME* <diskettenname>

Zweck: dient zum Umbenennen von Disketten.

Erläuterung: Eine neue Diskette erhält ihren Namen üblicherweise bei der Formatierung oder beim Kopieren einer anderen Diskette. Die dafür »zuständigen« Befehle sind *format* und *diskcopy* beziehungsweise ihre Äquivalente *Initialize* und *Duplicate* auf der Workbench. Möchte man nachträglich den Namen einer Diskette ändern, so ist dies im CLI nur mit *relabel* möglich.

Der Befehl *relabel* benötigt zwei Parameter. Der erste, <name>, ist der Name eines angeschlossenen Diskettenlaufwerks (meist wohl df0: oder df1:), in dem eine Diskette liegt, oder der alte Name einer Diskette. Der zweite Parameter <diskettenname> wird durch das Schlüsselwort NAME eingeleitet und ist der Name, den diese Diskette im angesprochenen Laufwerk tragen soll. Falls einer der beiden Namen Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden.

Beispiele:

```
1>relabel df0: WBDisk
1>relabel SYS: WBDisk
1>relabel "A500 WB 1.2 D:" WBDisk
```

Alle diese drei Befehle tun dasselbe (wenn im internen Laufwerk die Startdiskette liegt und den Namen *A500 WB 1.2 D* trägt). Sie benennen die Diskette im internen Laufwerk um in *WBDisk*.

Dateien und Directories werden übrigens mit dem Befehl *rename* (siehe unten) umbenannt.

rename

Format: rename *FROM* <alter name> *TO/AS* <neuer name>

Zweck: dient zum Umbenennen von Dateien und Directories.

Erläuterung: Der Befehl *rename* gibt einer Datei oder einem Directory einen neuen Namen. Er hat zwei Parameter: <alter name>, den alten und den neuen Namen der Datei/des Directory und <neuer name>, den neuen Namen der Datei/des Directory. Falls einer der beiden Namen Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden. Weiterhin darf noch keine Datei namens <neuer name> existieren, sonst tut *rename* nichts und liefert nur eine Fehlermeldung.

Mit *rename* kann man Dateien aber nicht nur im strengen Sinne umbenennen, sondern sie auch von einem Directory in ein anderes legen, sofern sich beide Directories auf derselben Diskette oder Festplatte befinden. Hierzu muß sich nur der erste Teil des vollständigen Pfadnamens in <alter name> und <neuer name> unterscheiden. Das wirkt dann, als würde <alter name> in ein anderes Directory kopiert und das Original dann gelöscht. (Ähnlich wie der mv-Befehl in UNIX.) Das zweite Directory muß allerdings vor Ausführung des Befehls schon existieren (siehe makedir).

Beispiele:

```
1>rename Text1 TO Text2
1>rename Text1 AS Text2
1>rename Text1 Text2
1>rename :Texte/Text1 AS :devs/printers/Text2
```

Die ersten drei Befehle tun dasselbe. Sie benennen die Datei *Text1* im aktuellen Directory um in *Text2*. Der letzte Befehl aber legt die Datei Text1 aus dem Directory *:Texte* unter dem Namen Text2 in das Directory *:devs/printers* auf der aktuellen Diskette.

Komplette Disketten oder Festplatten werden übrigens mit dem Befehl *relabel* (siehe oben) umbenannt.

run

Format: run <bel. Befehlsfolge>

Zweck: dient zum Ausführen eines oder einer Gruppe von CLI-Befehlen als separater Task (im »Hintergrund«).

Erläuterung: Wenn Sie den Befehl *run* an den Anfang einer Befehlszeile und dahinter einen weiteren kompletten CLI-Befehl setzen, so wird der letztere als separater Task abgearbeitet, der vom aktuellen CLI-Task unabhängig ist. Das spart Ihnen die Arbeit, zuvor mit *newcli* extra einen neuen CLI-Task dafür erzeugen zu müssen. Sofort nach dem Befehl *run* können Sie dem aktuellen CLI einen neuen Befehl erteilen, während nebenher der andere Befehl abläuft. Man nennt dies »Ausführen eines Befehls im Hintergrund«. Die Bezeichnung »im Hintergrund« kommt daher, daß dieser neue Task nicht mit dem Benutzer kommunizieren kann, da er kein eigenes Fenster besitzt. Er bleibt also im Hintergrund Ihrer Aufmerksamkeit, während Sie sich vordringlich (»im Vordergrund«) mit anderen Dingen beschäftigen.

Nur Meldungen von diesem anderen Task erscheinen in dem CLI-Fenster, von dem aus er gestartet wurde. Das kann dann verwirrend sein, wenn zwei oder mehr Tasks gleichzeitig ins Fenster schreiben. So können zum Beispiel während der Ausführung eines *dir*-Befehls zwischen den aufgelisteten Dateinamen Meldungen eines Hintergrund-Task auftauchen. Hat man gar mehrere Tasks im Hintergrund laufen, die Ausgaben ins CLI-Fenster schicken, ist das Chaos geradezu vorprogrammiert. Üblicherweise wird man deshalb wohl dafür einen CLI-Befehl verwenden, der längere Zeit zu seiner Ausführung benötigt und kein weiteres Eingreifen von seiten des Benutzers nötig hat und auch keine Meldungen ausgibt.

Der Befehl *run* kennt keine Parameter im eigentlichen Sinn des Wortes. Der Rest der Zeile hinter dem Wort *run* muß ein gültiger CLI-Befehl sein, der dann von *run* im Hintergrund ausgeführt wird. Dieser Befehl kann sich sogar über mehrere Zeilen ausdehnen, indem man die Möglichkeit der Zeilenfortführung nutzt. Hierzu gibt man am Zeilenende ein Pluszeichen (+) ein und drückt dann auf [Return]. Erst nach dem ersten [Return] ohne ein Pluszeichen davor beginnt die Abarbeitung der so erstellten Liste von Befehlen.

Beispiele:

```
1>run dir
```

Dieser Befehl ist ein schlechtes Beispiel für die Anwendung von *run*. Er läßt den *list*-Befehl mit vielen Bildschirmausgaben im Hintergrund laufen.

Der folgende Befehl ist hingegen recht sinnvoll. Er schickt den Inhalt einer Datei auf den Drucker, nachdem er sie vorher kopiert hat, löscht die Kopie nach Beendigung des Druckvorgangs und meldet dann, daß er fertig ist:

```
1>run copy BriefRobert druckvorlage +
copy druckvorlage PRT: +
delete druckvorlage +
echo "Druckvorgang beendet !"
```

Achten Sie bei der Verkettung mehrerer Zeilen darauf, daß unmittelbar hinter dem Fortführungssymbol + sofort [Return] getippt werden muß, ansonsten gilt die folgende Zeile als neue Zeile und nicht als Fortführung der vorangehenden.

say

Format: say <text>

Zweck: dient zur Ausgabe eines (meist kurzen) Textes als synthetische Sprache.

Erläuterung: Der Befehl *say* ist das akustische Gegenstück zum Befehl *echo*. Er nutzt die Fähigkeit des Amiga zur Sprachsynthese und gibt den Text, der als Parameter übergeben wurde, über einen angeschlossenen Lautsprecher aus. Die Aussprache ist nicht gerade natürlich zu nennen, aber doch verständlich. Wenn kein Lautsprecher an einen oder beide der Stereo-Ausgänge hinten am Gehäuse des Amiga 500 angeschlossen ist, sind solche Meldungen allerdings nicht zu hören. Der CLI-Befehl *say* ist übrigens nichts anderes als das Werkzeug *Say*, das Sie schon kennengelernt haben. Sie können in <text> auch dieselben Befehle zur Veränderung der Stimmlage, Sprechgeschwindigkeit und so weiter verwenden, die für diesen Zweck im Kapitel *Die Werkzeuge der Workbench* beschrieben wurden. Zusätzlich dazu können Sie auch noch die Option *-x* mit einem Dateinamen dahinter verwenden. *Say* spricht dann den gesamten Inhalt dieser Datei aus, als hätten Sie ihn als Parameter eingegeben.

Eine Verwendung von *say* bietet sich vor allem innerhalb von Kommandofolgen an. So könnte die Beendigung einer Kommandofolge zum Beispiel statt mit einer Bildschirmmeldung, die man vielleicht übersehen würde, durch ein lautes »Fertig!« kundgetan werden.

Beispiele:

```
1>say Hello
1>say -x s:startup-sequence
```

Der erste Befehl spricht nur das Wort »Hello«, der zweite den gesamten Inhalt der Textdatei *startup-sequence* im logischen Laufwerk *S:* aus.

search

Format: search *FROM* <name>* SEARCH <suchtext> *ALL*

Zweck: dient zur Suche nach einem bestimmten Text in einer beliebigen Gruppe von Dateien.

Erläuterung: Falls Sie einmal in die Verlegenheit kommen, nicht mehr genau zu wissen, in welchen Dateien Sie sich auf bestimmte Begriffe oder Namen bezogen haben, so hilft Ihnen der *search*-Befehl. Er erlaubt es, eine Gruppe von Dateien nach dem Text zu durchsuchen, der hinter dem Schlüsselwort SEARCH steht. (Falls dieser Text Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden.)

Jede Zeile in jeder durchsuchten Datei, die diesen Text enthält, wird ausgegeben. (Diese Liste kann natürlich mittels > auch in eine Datei statt auf den Bildschirm geschickt werden.) Beim Vergleich des Suchtextes mit dem Inhalt der Datei wird Groß- oder Kleinschreibung nicht berücksichtigt. (Mit Umlauten hat diese Suchmethode allerdings Schwierigkeiten: ein großes Ö ist so zum Beispiel verschieden von einem kleinen ö.) Der *search*-Befehl ist vor allem für Programmierer gedacht, die in ihren Programmen nach bestimmten Dingen (Prozeduren, Variablen, und so weiter) suchen. Er kann unter Umständen auch für den »normalen« Anwender von Nutzen sein.

Geben Sie für <name> den Namen eines Directory ein, so werden alle Dateien in diesem Directory durchsucht. Wenn Sie am Ende der Zeile noch das Schlüsselwort ALL folgen lassen, erstreckt sich die Suche auch auf alle Dateien in Directories, die dem durchsuchten hierarchisch untergeordnet sind (in diesem liegen). Statt ein Directory durchsuchen zu lassen, können Sie die zu berücksichtigenden Dateien auch in Form eines Musters angeben. Statt des Namens eines Directory geben Sie für <name> einfach das Muster an. Nur die Dateien, deren Namen in das Muster passen, werden dann durchsucht. Wird überhaupt kein Muster oder kein Directoryname angegeben, so werden die Dateien im aktuellen Directory durchsucht.

So werden zum Beispiel durch den Befehl *search df0: SEARCH »Sehr geehrt«* alle Dateien auf der Diskette im eingebauten Laufwerk nach dem Text »Sehr geehrt« durchsucht. Der Befehl search Brief#? SEARCH »Hallo!« hingegen durchsucht alle Dateien im aktuellen Directory, deren Name mit »Brief« beginnt, nach »Hallo!«.

Beim Durchsuchen der verschiedenen Dateien haben Sie die Möglichkeit, die Suche in der aktuellen Datei oder auch komplett abbrechen zu lassen. Halten Sie die [Ctrl]-Taste fest und tippen Sie gleichzeitig [D], so wird die Durchsuchung der gerade bearbeiteten Datei abgebrochen und gleich mit der nächsten fortgefahren. Halten Sie hingegen die [Ctrl]-Taste fest und tippen gleichzeitig [C], so wird die ganze Suche abgebrochen.

Beispiele:

```
1>search :texte SEARCH "Sehr geehrt"
1>search df0: "Sehr geehrt" ALL
```

Der erste Befehl sucht den Text »Sehr geehrt« nur im Directory *texte* auf der aktuellen Diskette. Der zweite Befehl klappert hingegen alle Dateien auf der Diskette im internen Laufwerk nach demselben Suchtext ab.

setclock

Format: setclock *OPT LOAD* *OPT SAVE*

Zweck: dient zum Stellen und Abfragen der in der Speichererweiterung Amiga 501 eingebauten Uhr.

Erläuterung: Wenn Sie Ihrem Amiga 500 eine Speichererweiterung Amiga 501 mit eingebauter Uhr spendiert haben, benötigen Sie den Befehl *setclock*. Eine Form des *setclock*-Befehls (*setclock opt save*) dient dazu, Amiga-DOS die Uhrzeit mitzuteilen, die die batteriegepufferte Uhr enthält. Die andere Form (*setclock opt load*) kann diese Uhr stellen. In beiden Fällen werden Uhrzeit und Datum auch im CLI-Fenster ausgegeben.

Beispiele:

```
1>setclock opt load
1>setclock opt save
```

Der erste Befehl bewirkt, daß Datum und Uhrzeit aus der batteriegepufferten Uhr ausgelesen und Amiga-DOS mitgeteilt werden. (Amiga-DOS weiß danach also, »wie spät es ist«.) Er ist bei jedem Neustart nötig, damit Sie Datum und Uhrzeit nicht selbst eingeben müssen und findet sich deshalb normalerweise auch in der Startup-Sequence. Der zweite Befehl stellt die batteriegepufferte Uhr auf die Uhrzeit und das Datum, das Amiga-DOS in diesem Moment für das/die richtige hält. Wenn Sie Datum und/oder Uhrzeit mit dem CLI-Kommando *date* oder dem Werkzeug *Preferences* verändert haben, müssen Sie *setclock opt save* aufrufen, damit auch die batteriegepufferte Uhr »davon erfährt«. Vergessen Sie das, vergißt der Amiga 500 im Gegenzug nach dem nächsten Neustart Ihre Änderungen.

setdate

Format: setdate *FILE* <name> *DATE* <datum> *TIME* *<zeit>*

Zweck: dient zur Änderung des Datums der letzten Modifikation an einer Datei, wie es der Befehl *list* zum Beispiel anzeigt.

Erläuterung: Sie werden den Befehl *setdate* sehr selten benötigen. Mit ihm können Sie nämlich das Datum der letzten Schreiboperation (Modifikation) bei einer Datei beliebig ändern. Wenn dieses Datum stimmt, also die letzte Änderung wirklich zu diesem Datum und dieser Uhrzeit vorgenommen wurde, gibt es keinen Grund, es nachträglich zu ändern.

Für die Datums- und Uhrzeitangaben <datum> und <zeit>, die Sie zusammen mit *setdate* verwenden, gelten dieselben Einschränkungen und Möglichkeiten wie beim *date*-Befehl. Zum Beispiel sind auch die Namen von Wochentagen und *tomorrow* sowie *yesterday* korrekte Datumsangaben.

Beispiele:

```
1>setdate Brief 10-jul-87 15:00:00
1>setdate Brief tomorrow
```

Der erste Befehl bewirkt, daß danach der zehnte Juli 1987 15 Uhr als Zeitpunkt der letzten Änderung an der Datei *Brief* gilt. Nach dem zweiten Befehl gilt die Uhrzeit 0 Uhr am Tag nach dem, an dem der Befehl eingegeben wurde, als Zeitpunkt der letzten Änderung an Briefe. Sie könnten sich mit Hilfe des Befehls *list Brief* davon überzeugen.

skip

Format: skip <marke>

Zweck: dient zum Überspringen einer Gruppe von CLI-Befehlen innerhalb einer Kommandofolge.

Erläuterung: Den Befehl *skip* (engl. *to skip = überspringen, übergehen*) werden Sie nie direkt an das CLI erteilen. Er ist nur in Kommandofolgen sinnvoll. Stößt das CLI in einer Kommandofolge auf den Befehl *skip <name1>*, so werden alle Befehle übersprungen, bis eine Zeile mit dem Befehl *lab <name2>* am Anfang gefunden wird und <name1> und <name2> gleich sind.

Mehr zum *skip*-Befehl aber im übernächsten Kapitel *Einfacher arbeiten mit Kommandofolgen*.

sort

Format: sort *FROM* <dateiname1> *TO* <dateiname2> *COLSTART* <*n*>

Zweck: dient zum Sortieren einer Textdatei.

Erläuterung: Mit dem Befehl *sort* (engl. *to sort = sortieren*) können Sie eine Textdatei zeilenweise sortieren. Eine Textdatei zeichnet sich ja dadurch aus, daß Sie nur druckbare Zeichen enthält und in Zeilen aufgeteilt ist, die durch ein besonderes Zeichen (den unsichtbaren Zeilenvorschub) getrennt sind. Hierbei enthält die Datei mit dem Namen <dateiname1> den unsortierten Text und die Datei mit dem Namen <dateiname2> nach Ausführung des Befehls den sortierten Text. Die Sortierung geschieht alphabetisch, beginnend mit der ersten Spalte jeder Zeile, und ignoriert Unterschiede zwischen Groß- und Kleinschreibung. (Mit Umlauten hat diese Sortiermethode allerdings Schwierigkeiten: ein großes Ö wird zum Beispiel verschieden von einem kleinen ö angeordnet.) Wollen Sie, daß bei der Sortierung der Text in jeder Zeile bis zu einer bestimmten Spalte ignoriert wird, so können Sie die Nummer dieser Spalte hinter dem Schlüsselwort COLSTART angeben.

Der Befehl *sort* ist ein sehr primitives Sortierprogramm, das nur für die wenigsten Anwendungsfälle geeignet sein dürfte. Normalerweise kann es nur Dateien sortieren, die maximal 200 Zeilen enthalten. Bei größeren Dateien müssen Sie den Speicher, der dem Programm für diese Zwecke zur Verfügung steht, mit dem Befehl *stack* (siehe unten) erhöhen. Wie groß der Speicher für Dateien bestimmter Größe sein muß, ist dabei sehr stark Erfahrungssache. *Sort* ist deshalb eigentlich nur für erfahrene Benutzer und kleine Datenmengen geeignet!

Beispiele:

```
1>sort liste TO sortierteListe
1>sort Adressen TO nachPLZ COLSTART 20
```

Der erste Befehl bewirkt, daß die Zeilen der Textdatei *liste* sortiert werden und die sortierte Folge von Zeilen in der Datei *sortierteListe* abgelegt wird. Der zweite Befehl sortiert die Datei *Adressen* und legt die sortierten Zeilen in *nachPLZ* ab. Dabei werden für die Sortierung nur die Buchstaben in jeder Zeile berücksichtigt, die ab der zwanzigsten Spalte kommen. Enthält die Datei *Adressen* eine Adressenliste, in der jede Adresse eine Zeile einnimmt und die Postleitzahl in der zwanzigsten Spalte beginnt, haben Sie auf diese Weise die Adressen nach PLZ sortiert.

stack

Format: stack <*n*>

Zweck: dient zum Anzeigen und Ändern der Stack-Größe.

Erläuterung: Mit dem Befehl *stack* kann die Größe eines bestimmten Speicherbereichs (des sogenannten *Stacks* oder *Stapelspeichers*) geändert werden, die das CLI einem Programm (Befehl) zuteilt, bevor es gestartet wird. Dieser Speicherbereich dient im wesentlichen für den sich laufend ändernden (dynamischen) Speicherbedarf eines Programms.

Ohne tiefergehende Kenntnisse der Programmierung dürfte der *stack*-Befehl für Sie wenig brauchbar sein. Falls aber bei Anwendung des *sort*-Befehls einmal ein Fehler auftauchen sollte, so liegt dies wahrscheinlich an mangelndem Speicherplatz. Aber auch andere Programme können in solche Schwierigkeiten kommen. Verdoppeln Sie dann versuchsweise diesen Speicherplatz mittels *stack*.

Tippen Sie *stack* ohne Parameter ein, so wird Ihnen die aktuelle Stackgröße angezeigt. Sie beträgt normalerweise 4000 (Byte). Tippen Sie hinter *stack* noch eine Zahl <n> ein, so wird der für den Stack reservierte Platz <n> gesetzt. Seien Sie aber vorsichtig mit zu kleinen Werten für <n>. Viele Programme benötigen mindestens 4000 Byte (*stack 4000*).

status

Format: status *PROCESS <n> FULL TCB*

Zweck: dient zum Anzeigen von Informationen über die verschiedenen laufenden CLI-Tasks.

Erläuterung: Mit dem Befehl *status* können Sie jederzeit erfahren, was gerade in den verschiedenen CLI-Fenstern geschieht, die Sie zuvor geöffnet haben. Wenn Sie einfach *status* ohne Parameter angeben, erhalten Sie eine Liste der aktiven CLI-Fenster und der Kommandos, die im jeweiligen Fenster gerade laufen. Wenn Sie den Schalter TCB verwenden, wird zu jedem Task der *Task-Control-Block* ausgegeben. Dieser enthält neben der Task-Nummer Informationen über die Stackgröße (vergleichen Sie den CLI-Befehl *stack*) und die Priorität (vergleichen Sie den CLI-Befehl *changeTaskPri*). Der Schalter FULL bewirkt, daß sowohl diese Informationen gezeigt werden als auch die gerade in dem entsprechenden CLI-Fenster laufenden Programme genannt werden.

Diese Informationen werden normalerweise für alle aktiven CLI-Tasks aufgeführt. Wenn Sie jedoch unmittelbar hinter *status* (oder hinter dem Schlüsselwort PROCESS) eine Zahl <n> angeben, so werden nur Informationen für das CLI-Fenster mit der entsprechenden Task-Nummer angezeigt.

Beispiele:

```
1>status
1>status FULL
1>status 4 FULL
```

Der erste Befehl listet knappe Informationen über alle aktiven CLI-Tasks auf, während der zweite umfangreichere Informationen über diese Tasks liefert. Der letzte Befehl liefert ebenfalls den vollen Satz von Task-Informationen, jedoch nur zu dem CLI-Task mit der Nummer 4.

type

Format: type <dateiname> *TO <ausgdatei> OPT N OPT H*

Zweck: dient zum Ausgeben einer Textdatei (auf den Bildschirm)

Erläuterung: Der Befehl *type* entspricht weitgehend dem Befehl *copy*, wenn dieser eine Textdatei »auf den Bildschirm kopiert« (mit >*). Er ist allerdings etwas einfacher zu handhaben, wenn man die Datei auf dem Bildschirm ausgeben will (dann kann man sich den TO-Parameter sparen) und bietet gegenüber *copy* zwei zusätzliche Möglichkeiten. Die Schlüsselwörter OPT N sorgen dafür, daß vor jeder neuen Zeile die aktuelle Zeilen-Nummer ausgegeben wird. Die Option H bewirkt, daß die Datei nicht in Form von Text, sondern als Folge von sogenannten Hexadezimal-Zahlen ausgegeben wird. Sie können dann auch Dateien, die nicht nur Text enthalten, auf dem Bildschirm betrachten, was aber wohl nur für Programmierer oder sehr erfahrene Benutzer von Interesse ist.

wait

Format: wait *<n> SECS MINS UNTIL <uhrzeit>*

Zweck: dient zum Verzögern eines Befehls, entweder um eine gewisse Zeit oder bis zu einem bestimmten Zeitpunkt.

Erläuterung: Der Befehl *wait* (engl. *to wait = warten*) dürfte vor allem innerhalb von Kommandofolgen oder *run*-Befehlen sinnvoll sein. *Wait* verzögert die Ausführung eines nachfolgenden Befehls und ist deshalb für sich alleingenommen reichlich sinnlos. Er wird erst dann beendet, wenn eine Anzahl von Sekunden oder Minuten verstrichen ist, oder wenn eine bestimmte Uhrzeit erreicht wurde. Die letzte Möglichkeit ist natürlich nur dann sinnvoll einsetzbar, wenn die Amiga-DOS bekannte Uhrzeit korrekt ist (zum Beispiel mittels *date* oder *setclock* gestellt wurde).

Wait kann auch ohne Parameter aufgerufen werden. Er wartet dann eine Sekunde. Wird nur <n> angegeben und weder das Schlüsselwort SECS noch das Schlüsselwort MINS verwendet, wartet *wait* <n> Sekunden.

Beispiele:

```
1>wait
1>wait 10 secs
1>wait 5 mins
1>wait 11:11:00
```

Der erste Befehl (*wait* ohne jeden Parameter) wartet genau eine Sekunde. Der zweite wartet 10 Sekunden und der dritte fünf Minuten. Besonders exakt sind diese Verzögerungszeiten aber nicht, da das CLI auch eine gewisse Zeit benötigt, um das Programm *wait* zu finden, zu laden und zu starten. Der dritte Befehl hingegen ist relativ exakt, er wartet, bis die von Amiga-DOS registrierte Zeit (die nicht mit der in der batteriegepufferten Uhr übereinstimmen muß) gleich 11 Uhr 11 Minuten und 11 Sekunden ist.

why

Format: why

Zweck: dient zum Erläutern der Gründe für den Fehlschlag des vorangehenden CLI-Befehls.

Erläuterung: Wenn ein CLI-Befehl – aus welchem Grund auch immer – mißlang, so wird üblicherweise eine Fehlermeldung ausgegeben. Diese Fehlermeldung kann ausreichen, manchmal ist sie aber recht knapp oder besteht nur aus einer Zahl (eine Unsitte, deren Gründe weit zurück in der Urgeschichte der Computer, also vor 1980, liegen).

Tippt man sofort nach dem fehlgeschlagenen Befehl *why*, so bekommt man eine knappe Erläuterung der Fehlerursache in »Klartext«. Dieser Klartext ist aber oft mit so vielen Fachausdrücken geladen, daß er Ihnen auch nicht immer helfen kann. In diesem Fall hilft Ihnen aber vielleicht die Fehlerliste im Anhang dieses Buches.

Vergleichen Sie zu diesem Thema auch die Beschreibung des CLI-Befehls *fault,* der jederzeit – nicht nur unmittelbar nach einem Fehler – aufgerufen werden kann.

8.4 Überblick

Zum Abschluß dieses Kapitels folgt nun eine Liste mit allen CLI-Befehlen. Diese Liste dürfte meist für den täglichen Einsatz des CLI ausreichen. Falls Sie die genauen Parameter, die ein bestimmter Befehl hat, vergessen haben, so können Sie diese meist mit der sogenannten »Help-Funktion« erfahren. Tippen Sie dazu einfach den Namen des Befehls, dessen Parameter Sie vergessen haben, und ein Fragezeichen ein und drücken Sie auf [Return]. Die Namen der möglichen Parameter und die Eigenschaften (optional, nichtoptional, mit Schlüsselwort und so weiter) werden Ihnen dann gleich mitgeteilt.

Hinter den Parameter-Namen befinden sich manchmal zusätzlich noch die Codes /A, /K und /S. Der Code /A bedeutet, daß der Befehl den Parameter unbedingt benötigt (Parameter ohne /A sind optional). Der Code /K bedeutet, daß man beim Aufruf des Befehls vor dem eigentlichen Parameter noch das zugehörige Schlüsselwort angeben muß. Dies hilft dem CLI, einen Parameter von anderen zu unterscheiden. Der Code /S bedeutet, daß es sich um ein Schlüsselwort ohne Parameter, also um einen »Schalter« handelt. Dies dürfte als Gedächtnisstütze fast immer genügen.

Befehle für die Datei- und Diskettenverwaltung:

Befehl	Wirkung
cd	Ändert das aktuelle Directory.
copy	Kopiert eine Datei oder eine Gruppe von Dateien.
delete	Löscht eine Datei oder eine Gruppe von Dateien.
diskchange	Meldet einen Diskettenwechsel bei einem Laufwerk, das einen solchen Vorgang nicht automatisch dem Amiga-DOS mitteilt.
diskcopy	Kopiert komplette (nicht kopiergeschützte) Disketten.
dir	Listet den Inhalt eines Directory auf.
filenote	Versieht eine Datei mit einem bis zu 80 Buchstaben langen Kommentar.
format	Formatiert und initialisiert eine Diskette oder Festplatte.
install	Macht aus einer formatierten Diskette eine bootfähige (startfähige) Diskette.
list	Listet detaillierte Informationen über ein Directory oder eine Gruppe von Dateien auf.
makedir	Erzeugt ein neues Directory.
protect	Setzt die Schutzflags einer Datei (wodurch diese zum Beispiel vor unbeabsichtigtem Löschen geschützt wird).
relabel	Ändert den Namen einer Diskette.
rename	Benennt eine Datei um und/oder legt sie in ein anderes Directory.
setdate	Ändert Datum und Uhrzeit des Zeitpunkts, an dem eine Datei (angeblich) das letzte Mal modifiziert wurde.
type	Gibt eine Datei auf dem Bildschirm aus (auf Wunsch hexadezimal oder mit Zeilennummern).

Systembefehle:

Befehl	Wirkung
assign	Ändert logische Gerätezuordnungen im System (kann für Abkürzungen mißbraucht werden).
binddrivers	Aktiviert die Treiberdateien in der *Expansion*-Schublade.
break	Bricht ein Programm in einem anderen CLI-Fenster ab.
changetaskpri	Ändert die Priorität des CLI-Tasks gegenüber anderen gleichzeitig aktiven Tasks (Programmen).
date	Zeigt oder ändert das intern gespeicherte Systemdatum und die Systemzeit.
endcli	Beendet das CLI und schließt dessen Fenster.
format	Formatiert und initialisiert eine 3,5-Zoll-Diskette.
info	Liefert Überblicksinformationen über das System.
install	Macht aus einer formatierten Diskette eine bootfähige (startfähige) Diskette.
mount	Meldet ein zusätzliches Peripheriegerät mit seinem Namen beim Amiga-DOS an.
path	Ändert den aktuellen Suchpfad für Programme oder zeigt ihn an.
setclock	Stellt die batteriegepufferte Uhr in der Speichererweiterung Amiga 501 oder übergibt deren Uhrzeit an Amiga-DOS.
stack	Ändert die Größe des Stacks, der beim Start eines anderen Programms vom CLI reserviert wird (Standard = 4000 Byte).
newcli	Erzeugt ein neues CLI (als eigenen unabhängigen Prozeß).

Informative Befehle:

Befehl	Wirkung
fault	Zeigt den Fehlertext zu einer Fehlernummer an.
status	Zeigt verschiedene Informationen über andere Tasks, die im Moment geladen sind (nur andere CLI-Tasks).
why	Zeigt (als Text) die Ursache des letzten Fehlers, der sich bei der Ausführung eines Programms ereignet hat.
info	Listet eine Reihe von Informationen über die zur Verfügung stehenden Laufwerke und die darin befindlichen Disketten auf (zum Beispiel den noch freien Platz).

Hilfsbefehle (Utilities):

Befehl	Wirkung
diskdoctor	Repariert defekte Disketten / Festplatten und stellt versehentlich gelöschte Dateien wieder her.
ed	Ruft den Fullscreen-Texteditor auf.
edit	Ruft den Zeilen-Texteditor auf.
join	Verbindet bis zu 15 einzelne Dateien zu einer neuen Datei.
prompt	Ändert die Meldung, mit der das CLI meldet, daß es zum Empfang des nächsten Befehls bereit ist.
run	Läßt einen oder mehrere CLI-Befehle im Hintergrund ablaufen.
search	Durchsucht alle Dateien eines Directory nach einem bestimmten Text.
sort	Sortiert Textdateien alphabetisch nach der ersten oder einer beliebigen Spalte in jeder Zeile.

Befehle für Kommandofolgen:

Befehl	Wirkung
echo	Gibt einen Text am Bildschirm aus.
else	Leitet die zweite Hälfte einer *If-else-endif*-Gruppierung zur bedingten Ausführung eine Teilfolge von Befehlen ein.
endif	Beendet eine *If-else-endif*-Gruppierung zur bedingten Ausführung einer Teilfolge von Befehlen.
execute	Startet eine Kommandofolge.
failat	Legt fest, ob und bei welchen Fehlern Kommandofolgen fortgesetzt werden dürfen, wenn während der Ausführung einer Kommandofolge Fehler auftreten.
if	Leitet eine *If-else-endif*-Gruppierung zur bedingten Ausführung einer Teilfolge von Befehlen ein.
lab	Versieht eine Stelle in einer Kommandofolge mit einer Marke.
quit	Verläßt eine Kommandofolge »mittendrin« und gibt eventuell einen Fehlercode zurück.
say	Spricht einen Text aus.
skip	Springt (nur nach vorne) zu einer Marke.
wait	Wartet eine bestimmte Zeitspanne oder bis zu einem bestimmten Zeitpunkt mit der Ausführung des nächsten Befehls.

9 Texteditoren für das CLI

Eine Aufgabe, die – nicht nur im CLI – immer wieder anfällt, ist das Erzeugen und Ändern von Textdateien. Ob dies einfache Notizen, komplexe Programme oder Kommandofolgen für das CLI sind, immer wird ein Werkzeug benötigt, das diese Aufgabe so gut wie möglich unterstützt. Ein solches Werkzeug wird üblicherweise »Texteditor« genannt. Für den Amiga 500 gibt es bereits viele derartige Texteditoren. Einige davon erhalten Sie beim Kauf des Amiga 500 schon dazu: *Ed*, *Edit*, *MicroEmacs* und *Notepad*.

9.1 Textbearbeitung auf dem Amiga – ein Überblick

Ed ist ein Texteditor »in Reinkultur«, mit dem *reine Textdateien* behandelt werden können. Es handelt sich nicht um eine komplette Textverarbeitung, die man zum Beispiel für Bürozwecke einsetzen könnte, sondern nur um ein Hilfsmittel zum Bearbeiten von unstrukturierten Texten. (Im angloamerikanischen Bereich wird diese Unterscheidung auch im verbalen Bereich etwas deutlicher. Bei einem komfortablen Textverarbeitungssystem spricht man meist von einem *word processor* und bei einem einfachen Texteditor eben auch von einem *editor* oder *text editor*.) Ein Texteditor erkennt keine Seiten, Kapitel, Absätze und vielleicht nicht einmal Worte. Er erkennt nur einen Strom von Buchstaben, der in Zeilen aufgeteilt ist. In einem Texteditor können Texte nicht fettgedruckt oder unterstrichen werden. Es ist ein sehr simples Werkzeug, das selbst für Briefe etwas spartanisch ist, aber doch wichtig für die Aufbereitung von Texten, die als Eingaben für andere Programme dienen sollen, oder für die Betrachtung von Texten, die andere Programme erzeugen.

So bestehen zum Beispiel Kommandofolgen nur aus Buchstaben, die zu Zeilen gruppiert sind, und auch die Dateien, die entstehen, wenn man die Ausgabe eines CLI-Befehls mittels > auf eine Datei umleitet, sind meist einfache Textdateien.

Der Texteditor *ed* ist allerdings schon ein recht komfortabler Vertreter seiner Gattung. Es ist ein sogenannter *Bildschirm-Editor* (engl. *screen editor*), der uns einen Ausschnitt (ein Fenster)

aus dem bearbeiteten Text zeigt. In diesem Ausschnitt kann man mit den Pfeil- oder Cursor-Tasten rechts unten auf der Tastatur hin- und herfahren und an beliebigen Stellen Texte überschreiben, löschen oder ergänzen. Andere Texteditoren – die Vertreter der Gattung *Zeileneditor* (engl. *line editor*) zum Beispiel – machen die Bearbeitung eines Textes wesentlich umständlicher. Sie erlauben unter anderem nur die Bearbeitung einer aktuellen Zeile, und jeder Wechsel zu einer anderen Zeile ist mit speziellen Befehlen verbunden. Obwohl *ed* in mancher Hinsicht einiges zu wünschen übrig läßt, kann man mit ihm doch schon recht gut auskommen.

Sein Nachteil – zumindest für einen Amiga-Benutzer – ist, daß er nur über die Tastatur gesteuert wird und ihm Maus- und Menübefehle sozusagen unbekannt sind. Dies ist bei *MicroEmacs* anders. Dieser Texteditor ist eine leicht vereinfachte Version des extrem leistungsfähigen Editors *emacs*, der sich in der Computerwelt seit langem großer Beliebtheit erfreut. *MicroEmacs* kann allein über die Tastatur gesteuert werden, akzeptiert aber auch Menübefehle. Zu fast jedem Kommando, das über die Tastatur eingegeben wird, existiert zum Beispiel auch ein Menübefehl. Und auch zur Bewegung der Schreibmarke im Text kann die Maus eingesetzt werden. Für den, der sehr viel Textverarbeitung betreiben muß – zum Beispiel Programmierer – bietet *MicroEmacs* eine Fülle von Fähigkeiten, die weit über das von *ed* gebotene hinausgehen.

MicroEmacs hat nur einen kleinen Nachteil. Er wurde zu dem Zeitpunkt, als dieses Buch entstand, zwar mit dem Amiga 500 ausgeliefert. Es kann aber sein, daß dieser Zustand nicht immer anhalten wird. In diesem Fall müssen Sie *MicroEmacs* (er wird sich dann auf der *Amiga-Toolkit-Diskette* befinden) separat erwerben. Das ist aber eine Ausgabe, die sich für jeden, der intensiv mit eigenen Programmen und anderen Textdateien umgehen muß, bald bezahlt macht.

Neben *ed* und *MicroEmacs* gibt es noch eine Reihe weiterer Werkzeuge, die Sie zur Textverarbeitung benutzen können. Hierzu gehört der recht umständliche Zeileneditor *edit*, das Werkzeug *Notepad* und das *List*-Fenster von Amiga-BASIC. *Notepad* kann wirklich recht gut zum Editieren kleinerer Textdateien eingesetzt werden, während sich mit *edit* wohl nur die wenigsten Anwender anfreunden werden und das *List*-Fenster von Amiga-BASIC nur zur Bearbeitung von BASIC-Programmen wirklich gut geeignet ist.

9.2 Der tastengesteuerte CLI-Editor »ed«

Kommen wir nun zum ersten der obengenannten Editoren: dem CLI-Editor *ed*. Er war in den ersten Versionen (1.0 und 1.1) von Amiga-DOS der einzig brauchbare Editor für längere Texte. Inzwischen dürften ihm *MicroEmacs* und vielleicht auch *Notepad* der Workbench Version 1.2 deutlich den Rang abgelaufen haben. Es lohnt sich aber nach wie vor, sich ein wenig intensiver mit ihm zu beschäftigen, weil er einige wichtige Konzepte der Textverarbeitung sehr gut demonstriert und den großen Vorteil hat, sehr klein zu sein (unter 20 Kbyte).

9.2.1 Aufrufen und Verlassen von »ed«

Der Texteditor *ed* wird üblicherweise vom CLI aus aufgerufen. Das genaue Aufruf-Format lautet:

```
ed <dateiname> SIZE <n>
```

Dabei kann `<dateiname>` sowohl der Name einer neuen Datei sein, die Sie mit *ed* erzeugen wollen, als auch der Name einer schon vorhandenen, die Sie verändern oder nur betrachten wollen. <dateiname> ist nicht optional, muß also immer mit angegeben werden. SIZE ist ein optionaler Parameter. Wenn Sie dem Schlüsselwort SIZE eine Zahl folgen lassen, so bestimmt diese die *Puffergröße* von *ed.* Das ist die maximale Größe des Textes (in Buchstaben beziehungsweise Anschlägen), die Sie mit *ed* gerade noch bearbeiten können. Standardmäßig ist dieser Puffer 40 000 Anschläge groß – mehr als ausreichend für unsere ersten Versuche.

Starten Sie nun das CLI, falls Sie sich noch nicht darin befinden, und legen Sie ein neues Directory für die folgenden Versuche an:

```
1>makedir df0:texte
```

Sie können es nach Beendigung dieses Kapitels löschen – es wird aber nur ein paar tausend Byte groß werden und Ihren Diskettenplatz kaum einschränken. Das neue Directory machen Sie nun bitte zum aktuellen Directory:

```
1>cd df0:texte
```

Starten Sie nun *ed*, indem Sie den folgenden Befehl eingeben und dann natürlich mit [Return] beenden:

```
1>ed text1
```

Sie sollten sich nun in *ed* befinden. Lassen Sie die Tastatur einen Moment in Ruhe und betrachten Sie den Bildschirm. Ein neues Fenster hat sich geöffnet, das den gesamten Bildschirm füllt und in dessen linker oberer Ecke ein kleines helles Rechteck erscheint, die sogenannte »Schreibmarke« (auf englisch *Cursor*). Das *ed*-Fenster ist ein ganz normales Fenster und kann mit den üblichen Methoden verkleinert und dann verschoben werden. Probieren Sie dies nun und Sie werden sehen, daß dahinter das CLI-Fenster auftaucht und alle anderen Objekte der Workbench. Sie können dieses Fenster ergreifen und verschieben, verkleinern und vergrößern und andere Programme starten (sofern noch genügend Speicher dafür vorhanden ist). Sie können aber keine Befehle in das CLI-Fenster eintippen, von dem aus Sie *ed* aufgerufen haben. Sie können zwar Befehle eingeben und diese erscheinen auch am Bildschirm. Mit der Ausführung wird aber erst dann begonnen, wenn Sie *ed* verlassen. Da Sie direkt *ed* aufgerufen haben, wartet das CLI jetzt erst auf die Beendigung von *ed*, bevor es neue Befehle akzeptiert. Wollen Sie den Editor aufrufen und trotzdem noch die Möglichkeit haben, das CLI zu nutzen, so müssen Sie den »Umweg« über den *run*-Befehl wählen. Mehr zu diesem Thema finden Sie bei der Beschreibung des *run*-Befehls im vorangehenden Kapitel. Vergrößern Sie das *ed*-Fenster nun wieder auf angenehme Größe (es muß nicht die volle Bildschirmgröße sein), damit Sie mit Ihren ersten Versuchen beginnen können.

Klicken Sie sicherheitshalber einmal mit der Maus ins Innere des *ed*-Fensters und tippen danach ein paar Buchstaben ein. Es kann ruhig ein sinnvoller Satz sein, Sie können aber natürlich auch einfach mit den Fingern auf die Tastatur trommeln. Die Buchstaben erscheinen nun nacheinander – in der Reihenfolge, in der Sie sie eingetippt haben – auf dem Bildschirm. Der Cursor, das helle Rechteck, wandert dabei immer weiter und steht immer rechts neben dem letzten eingegebenen Buchstaben. Drücken Sie nun ein paarmal auf die [Backspace]-Taste in der oberen rechten Ecke der Tastatur (falls Sie es vergessen haben, es ist die Taste [<-] mit dem Pfeil nach links darauf). Sie sehen dann, wie die letzten Buchstaben, die Sie getippt haben, gelöscht werden und der Cursor wieder nach links wandert.

Bis hierhin ist die Eingabe von Text genauso, wie Sie es aus dem CLI kennen. Eingetippter Text erscheint unter dem Cursor, der dadurch nach rechts wandert. Mit der [Backspace]-Taste kann jeweils der letzte Buchstabe rechts gelöscht werden.

Nun brechen Sie bitte Ihre erste *ed*-Sitzung einmal ab und kehren Sie zurück ins CLI. Sie wissen ja, es gibt zwei wichtige Dinge, die man sofort über jedes Programm lernen muß: wie man hineinkommt und wie man unbeschadet wieder herauskommt. Wie man hineinkommt, haben Sie schon gesehen, und nun werden Sie sehen, wie man wieder herauskommt.

Das Programm *ed* wird über einen sogenannten *Escape-Befehl* verlassen, von denen Sie weiter unten noch einige mehr kennenlernen werden. Tippen Sie zunächst kurz auf die [Esc]-Taste ([Esc] steht für *escape*) am oberen linken Rand der Tastatur. Der Cursor springt daraufhin an den unteren Rand des Fensters und erwartet eine weitere Eingabe. (Dieses Verhalten ist bei allen Escape-Befehlen gleich.) Tippen Sie nun den Buchstaben *X* und dann [Return]. Es erscheint nun kurz eine Meldung am unteren Fensterrand, die wahrscheinlich aber zu schnell verschwindet, als daß Sie sie lesen könnten. Danach verschwindet das *ed*-Fenster, und Sie sind wieder im CLI.

Tippen Sie nun den Befehl *dir* – und wie Sie sehen, haben Sie soeben eine neue Datei erzeugt, die unter dem Namen *text1* aufgelistet wird. Diese Datei werden wir nun im nächsten Kapitel weiter bearbeiten.

9.2.2 Die direkten ed-Befehle

Die Befehle, die *ed* für die Bearbeitung, das Abspeichern und Lesen von Texten kennt, zerfallen in zwei große Gruppen. Die *direkten* Befehle werden sofort ausgeführt, nachdem Sie auf eine bestimmte Taste oder eine Kombination von zwei Tasten gedrückt haben. Die *erweiterten* oder *Escape-Befehle* werden eingeleitet durch ein Drücken der [Esc]-Taste und erwarten danach immer eine weitere Eingabe, bevor der Befehl nach dem Drücken der [Return]-Taste endlich gestartet wird. In diesem Abschnitt werden die direkten Befehle vorgestellt, die Sie wahrscheinlich auch am häufigsten benötigen werden, und im folgenden Abschnitt dann die *Escape-Befehle*.

Starten Sie nun bitte erneut *ed* mit dem Befehl *ed text1*. Diesmal ist das erscheinende Fenster nicht leer, sondern zeigt Ihnen bereits den Text, den Sie bei der ersten Sitzung eingegeben haben. Dies illustriert sehr schön das unterschiedliche Verhalten von *ed*, je nachdem, ob Sie eine bereits vorhandene Datei editieren oder eine neue Datei erzeugen. Verlassen Sie *ed*

nämlich in der oben vorgestellten Weise ([Esc] [X]), wird der Text, den Sie bearbeitet haben, immer in eine Datei geschrieben, deren Namen Sie *ed* beim Aufruf als erstem Parameter mitgeteilt haben. Existierte bereits eine Datei dieses Namens, wird ihr alter Inhalt dabei zerstört. Gab es noch keine Datei dieses Namens, so wird eine neue unter diesem Namen erzeugt und in ihr der Text abgelegt.

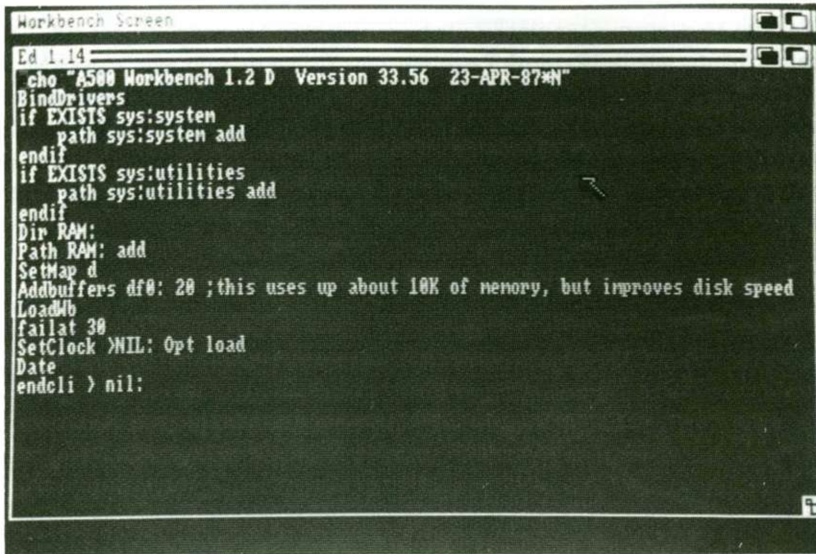


Bild 9.1: Der Texteditor ed in Aktion

Zwischendurch, während Sie einen Text bearbeiten, wird nur der »Pufferinhalt« geändert, der im RAM-Speicher des Amiga 500 verwaltet wird. Die Datei auf der Diskette bleibt unbeeinträchtigt. Sie brauchen sich also erst am Ende einer Sitzung zu überlegen, ob Sie Ihre Änderungen wirklich abspeichern oder lieber die alte Version eines Textes behalten wollen. Auch können Sie jederzeit während der Textbearbeitung zur alten Version zurückkehren. Nach diesen hoffentlich beruhigenden Worten nun aber zu den direkten Befehlen.

Fast jede der Tasten des Amiga 500 ist strenggenommen ein solcher direkter Befehl. Die Buchstaben-Tasten sind zum Beispiel ein Befehl, an der Stelle, an der sich gerade der Cursor befindet, einen Buchstaben einzufügen. Die [Return]-Taste ist ein Befehl, eine neue Zeile zu beginnen und so weiter. Direkte Befehle zerfallen in die folgenden Gruppen:

- Bewegung des Cursors
- Einfügen von Text
- Löschen von Text
- Verschieben des Textfensters im Gesamttext
- Besondere Befehle

Bewegung des Cursors

Die wichtigsten Befehle, die Sie innerhalb von *ed* nahezu andauernd benötigen werden, sind die Befehle für die Bewegung des Cursors. Hierzu dienen vor allem die vier mit einem Pfeil versehenen Tasten im rechten Drittel der Tastatur (links neben dem numerischen Block). Dies sind die sogenannten *Cursor-Tasten*. Sie bewegen die Schreibmarke (Cursor) um jeweils eine Position in der angegebenen Richtung. Versuchen Sie es einmal, um damit vertraut zu werden.

Falls Ihr Text im Moment nur aus einer Zeile besteht, tippen Sie einfach mehr ein. Tippen Sie dabei vor allem häufig auf die [Return]-Taste, um neue Zeilen zu beginnen. Tippen Sie so lange Text ein, bis das Fenster wenigstens gefüllt ist, damit Sie genug für die folgenden Experimente haben. Falls Ihnen kein passender Text einfällt, tippen Sie einfach eine Buchseite oder einen Zeitungsartikel ab. Sobald Sie am unteren Ende des *ed*-Fensters angekommen sind (machen Sie es ruhig etwas kleiner, damit Sie den Effekt schneller zu sehen bekommen), rutscht der bisher getippte Text nach oben, wie auch im CLI-Fenster. (Im Gegensatz zum CLI-Fenster kann man ihn aber wieder zurückholen, wie Sie weiter unten noch sehen werden.)

Sie sollten nun genügend Text für die meisten der folgenden Versuche haben. Experimentieren Sie nun ein wenig mit den Cursor-Tasten. Halten Sie auch einmal eine Taste etwas länger fest und beobachten, was geschieht. Nach einer kleinen Pause »wiederholt sich die Taste« nämlich von selbst und der Cursor schießt in die angegebene Richtung, als hätten Sie sehr oft nacheinander auf die Taste gedrückt. Dies funktioniert bei allen Cursor-Tasten und ist sehr praktisch, sobald man damit umgehen kann. Die Länge der Pause vor Beginn der Tasten-Wiederholung und die Geschwindigkeit der Wiederholung können übrigens mit dem Programm *Preferences* eingestellt werden.

Wenn der Cursor an das rechte Ende des Fensters kommt (machen Sie es ruhig etwas kleiner, damit Sie den Effekt schneller sehen), rutscht der ganze Text nach links. Auf diese Weise können Sie auch Texte bearbeiten, die breiter sind als das Fenster werden kann. *Ed* kann Texte bearbeiten, in denen die Zeilen eine maximale Breite von 255 Buchstaben haben. Das dürfte für die meisten Zwecke ausreichen. Um wieder zurückzukommen, um also den Text am linken Rand wieder sehen zu können, halten Sie einfach die Cursor-Taste fest, die nach links zeigt. Sobald der Cursor am linken Fensterrand ankommt, verschiebt sich der Fensterinhalt nämlich wieder nach rechts – allerdings nur so lange, bis Sie in der ersten Spalte angekommen sind. Genauso können Sie übrigens auch den Text nach oben und unten verschieben, indem Sie die nach oben und nach unten weisenden Cursor-Tasten verwenden. Probieren Sie es ruhig aus.

Dieses Verschieben des Textes im Fenster funktioniert allerdings nur, wenn Ihr Text mehr Zeilen hat, als im Fenster Platz haben. Das Fenster ist also wirklich ein Fenster in einem Text, das einen beliebigen Ausschnitt der Gesamtheit zeigen kann. Die folgende Abbildung versucht, dies bildlich zu verdeutlichen.

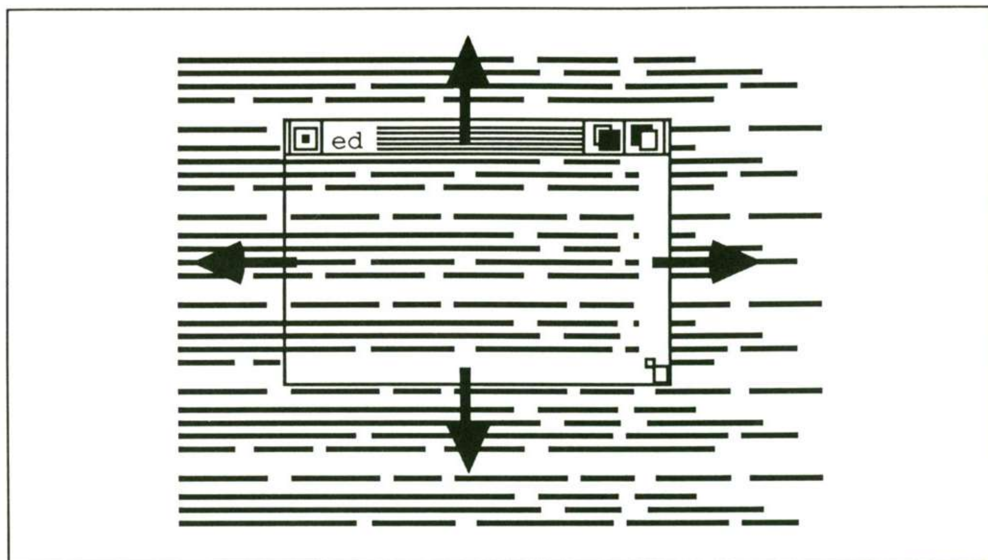


Bild 9.2: Ein langer Text und der Ausschnitt, den ed davon zeigt

Nun aber genug des Hin- und Herhüpfens im Text. Es gibt noch einige spezielle Befehle, die Sie zu »herausragenden« Stellen in einem Text bringen. Die Tastenkombination [Ctrl]+»]« bringt Sie stets an das Ende der Zeile, in der sich der Cursor gerade befindet. Wenn der Text breiter ist als das Fenster, verschiebt er sich dabei natürlich nach links. Falls sich der Cursor schon auf dem letzten Buchstaben der Zeile befand, springt er bei [Ctrl]+»]« zum Anfang der Zeile.

Ein Wort zur Schreibweise für solche Befehle: [Ctrl], dann ein Bindestrich und dahinter ein Buchstabe in eckigen Klammern oder Anführungsstrichen bedeutet, daß Sie die Taste am linken Rand der Tastatur festhalten sollen, auf der *Ctrl* steht, und gleichzeitig die Taste mit dem anderen Buchstaben drücken. Das Pluszeichen bedeutet, daß Sie beide Tasten gleichzeitig drücken müssen. (Fehlt ein Bindestrich bei ähnlichen Beschreibungen, müssen die beiden Tasten nacheinander gedrückt werden.) [Ctrl]+»]« bedeutet also Festhalten der [Ctrl]-Taste und Drücken der Taste »]« (eckige Klammer zu).

Die Tastenkombination [Ctrl]+[E] bringt Sie zum Anfang der ersten Zeile auf dem Bildschirm. Falls der Cursor aber schon dort war, springt er bei [Ctrl]+[E] zum Ende der letzten Zeile auf dem Bildschirm.

Die Tastenkombination [Ctrl]+[T] bringt Sie zum Anfang des nächsten Wortes am Bildschirm. [Ctrl]+[T] bringt Sie auf das Leerzeichen, das dem vorangehenden Wort folgt. Falls diese beiden Positionen außerhalb des Fensters liegen, wird der Text entsprechend verschoben.

Die [Tab]-Taste schließlich (auf halber Höhe am linken Rand der Tastatur) springt immer um N Schritte nach rechts. Die Zahl N, also die Schrittweite, ist durch einen anderen Befehl

einstellbar. Nach dem Start sind es drei Schritte. Wer schon andere Textverarbeitungssysteme kennt, wird die [Tab]-Taste zunächst vielleicht mißverstehen. Die [Tab]-Taste auf dem Amiga springt N Buchstaben weiter, fügt aber keine Buchstaben (Leerzeichen oder ein besonderes Tab-Zeichen) in den Text ein.

Damit wären die Tasten zur Bewegung der Schreibmarke schon aufgezählt. Spielen Sie ruhig ein wenig mit den Möglichkeiten herum. Weiter unten werden Sie noch andere Möglichkeiten für »größere Sprünge« im Text kennenlernen, die aber in eine andere Kategorie fallen. Falls Ihnen das Gelernte erst einmal reicht, beenden Sie die Sitzung einfach mit [Esc] [X] [Return].

Einfügen von Text

Beim Eintippen des Textes haben Sie bestimmt das eine oder andere Mal einen Fehler gemacht. Fällt er sofort auf, können Sie einen solchen Fehler ganz gut mit der [Backspace]-Taste beseitigen. Dies lohnt aber kaum, wenn er schon zwei Zeilen zurückliegt. Für die Korrektur solcher Fehler und größere Änderungen an Texten müssen Sie Texte löschen und nachträglich neue Texte einfügen können. Das werden Sie nun lernen. Zunächst kommt das Einfügen dran; das Löschen von Text wird im nächsten Abschnitt behandelt.

Falls Sie sich nicht noch im *ed* befinden, starten Sie ihn nun wieder mit *ed text1*. Sie bekommen dann wieder den Beginn des im letzten Kapitel fertiggestellten Textes gezeigt und können dort weitermachen, wo Sie bei der letzten Sitzung aufgehört haben. Gehen Sie nun mit den Cursor-Tasten oder anderen Bewegungsbefehlen an eine beliebige Stelle im Text und tippen einige neue Wörter ein. Wie Sie sehen, bewegt sich der alte Text rechts der Position der Schreibmarke beim Tippen nach rechts, um den neu eingegebenen Buchstaben Platz zu machen. Die meisten Textverarbeitungssysteme und Editoren arbeiten so. Neuer Text überschreibt nicht den alten, sondern wird zwischen zwei Buchstaben eingesetzt. Wollen Sie also ein Wort gegen ein anderes ersetzen, so müssen Sie das alte zuerst löschen und dann das neue einfügen!

Die Zeilen, die *ed* bearbeiten soll, können maximal 255 Buchstaben lang werden. Dabei ist es egal, ob Sie dieses Limit beim Eintippen einer neuen Zeile oder beim Einfügen von Text in eine bestehende Zeile überschreiten. Bei längeren Zeilen erfolgt eine Fehlermeldung (*Line to long*), die in der untersten Zeile des *ed*-Fensters erscheint. Betätigen Sie statt eines normalen Buchstabens ein [Return], so beginnt eine neue Zeile – wie Sie es ja auch schon kennen. Der Cursor springt aber nicht einfach in die nächste Zeile, sondern er fügt eine neue leere Zeile zwischen der aktuellen und der nächsten ein. Sie haben dies am Anfang vielleicht nicht bemerkt, weil hinter der Zeile, die Sie gerade eingegeben haben, keine andere mehr kam. Sie können auf diese Weise auch eine alte Zeile in zwei neue aufspalten, indem Sie den Cursor in die alte Zeile an die Stelle bewegen, an der getrennt werden soll und dann [Return] tippen.

Das waren aber schon alle »Befehle« für das Einfügen von Text. Nachdem wir bisher kreativ waren und nur neuen Text erzeugt haben, werden wir uns nun mit der Zerstörung – dem Löschen von Text – beschäftigen.

Löschen von Text

Einen der wichtigsten Befehle zum Löschen von Text haben Sie ja bereits kennengelernt: die [Backspace]-Taste. Diese Taste löscht den Buchstaben links von der Schreibmarke und bewegt

dann die Schreibmarke um eine Position nach links. Das funktioniert nicht nur am Ende einer Zeile, sondern auch mitten in einer Zeile. Der Text rechts vom gelöschten Buchstaben (falls vorhanden) rutscht nämlich beim Löschen eines Buchstabens ebenfalls um eine Position nach links. Probieren Sie es aus!

Mit der [Del]-Taste (rechts neben der [Backspace]-Taste) können Sie den hervorgehobenen Buchstaben, »auf« dem die Schreibmarke steht, löschen. Die Schreibmarke bleibt dabei an derselben Position stehen, nur der Text rechts davon rückt nach links auf.

Die Tastenkombination [Ctrl]+[O] (der Buchstabe »O«; nicht die Zahl Null) löscht schon etwas größere »Brocken«. Ihre Wirkung ist abhängig von dem Buchstaben, auf dem die Schreibmarke steht. Ist dieser Buchstabe ein Leerzeichen, so werden dieses Leerzeichen und alle folgenden Leerzeichen bis hin zum nächsten »Nicht-Leerzeichen« gelöscht. Befindet sich kein Leerzeichen an der Cursorposition, löscht [Ctrl]+[O] alle Buchstaben ab der Cursorposition bis zum nächsten Leerzeichen. Man könnte dazu auch sagen, daß [Ctrl]+[O] den Rest des Wortes löscht, in dem sich die Schreibmarke gerade befindet.

Die Tastenkombination [Ctrl]+[Y] löscht den »Rest« der Zeile rechts neben der Schreibmarke und den Buchstaben, auf dem die Schreibmarke steht.

Die Tastenkombination [Ctrl]+[B] hingegen löscht die ganze aktuelle Zeile, egal wo die Schreibmarke innerhalb der Zeile steht. Die darauffolgenden Zeilen rutschen dabei um eine Zeile hoch.

Verschieben des Text-Fensters im Text

Wie bereits erwähnt, zeigt *ed* immer nur einen Ausschnitt des gesamten bearbeiteten Textes. Er kann sich sowohl horizontal als auch vertikal weit über die Größe des Fensters hinaus erstrecken. Bei verschiedenen Befehlen wird automatisch dieser Ausschnitt verändert – so zum Beispiel, wenn Sie mit der Schreibmarke an den Rand des Fensters stoßen. Es gibt aber auch Befehle, mit denen man den Text ganz explizit im Fenster verschieben kann. [Ctrl]+[D] und [Ctrl]+[U] sind zwei Tastenkombinationen, mit denen der Text im Fenster gleich ein ganzes Stück nach unten (engl. *down*) bzw. nach oben (engl. *up*) bewegt werden kann.

Die Tastenkombination [Ctrl]+[D] bewegt den Text im Fenster, falls möglich, um 12 Zeilen nach unten (oben erscheinen also die 12 vorangehenden Zeilen) und bewegt gleichzeitig die Schreibmarke 12 Zeilen nach oben.

Die Tastenkombination [Ctrl]+[U] bewegt den Text im Fenster, falls möglich, um 12 Zeilen nach oben (unten erscheinen also die 12 folgenden Zeilen), und bewegt gleichzeitig die Schreibmarke 12 Zeilen nach unten.

Besondere Befehle

Zwei weitere direkte Befehle müssen noch erwähnt werden.

Die Tastenkombination [Ctrl]+[G] wiederholt den letzten Escape-Befehl, der erteilt wurde. Escape-Befehle sind ja typischerweise viel umständlicher einzugeben, weshalb dieser Befehl

einige Tipparbeit sparen kann. Wie sinnvoll das sein kann, erfahren Sie in einem der nächsten Abschnitte.

Die Tastenkombination [Ctrl]+[F] ändert die Groß-/Kleinschreibung des Buchstabens an der Cursorposition und verschiebt den Cursor um eine Position nach rechts. Leerzeichen und Sonderzeichen werden nicht beeinflußt (Umlaute leider auch nicht). Dies kann besonders praktisch für Programmierer sein, ist manchmal aber auch für die Fehlerkorrektur von Nutzen. Ich selbst pflege zum Beispiel beim schnellen Tippen häufiger Groß- und Kleinbuchstaben zu verwechseln.

9.2.3 Escape-Befehle

Die zweite Gruppe von Befehlen, die *ed* erkennt, ist etwas umständlicher zu erteilen. Einen davon, den Befehl zum Verlassen von *ed*, haben Sie ja schon kennengelernt. Alle diese Befehle verlangen mehrere Tastendrücke, können dafür aber auch einiges mehr als die direkten Befehle. Zusätzlich haben fast alle direkten Befehle noch einmal ein Gegenstück als Escape-Befehl, der im wesentlichen dasselbe tut. Dies ergibt erst im Zusammenhang mit der Möglichkeit, Befehle automatisch mehrmals hintereinander auszuführen, einen Sinn. Auf diese Möglichkeit wird im vorletzten Abschnitt dieses Kapitels noch etwas näher eingegangen.

Die Beschreibung der meisten Escape-Befehle wird knapp ausfallen. Eine ausführliche Erörterung verbietet sich aus Platzgründen. Nicht alle Leser dieses Buches würden sie schließlich auch zu schätzen wissen, da sie all diese Befehle vielleicht nie nutzen werden. Ein wenig mit den Befehlen zu experimentieren, die Sie wirklich interessieren, ist die beste Möglichkeit, damit vertraut zu werden (natürlich immer auf der Kopie der Original-Workbench-Diskette, die Sie zu Beginn dieses Buchteils ja anfertigen sollten).

Grundsätzliches zu Escape-Befehlen

Escape-Befehle werden immer durch einen Druck auf die [Esc]-Taste (Escape-Taste) eingeleitet. Die Schreibmarke springt dann in die unterste Zeile des Fensters, wo Sie hinter einem Sternchen (*) einen Befehl eingeben können. Nach der Ausführung des Befehls springt die Schreibmarke wieder zurück in den Text – eventuell an die Stelle, an der Sie [Esc] gedrückt haben, unter Umständen aber auch an eine andere. Diese Zeile am unteren Rand des Fensters heißt *Kommandozeile* oder *Befehlszeile* und dient der Eingabe von Escape-Befehlen, aber auch für die Mitteilung von Fehlermeldungen.

Escape-Befehle bestehen aus einem oder zwei Buchstaben, worauf eventuell noch ein oder zwei Parameter in Form eines Textes folgen. Ob Sie den Befehl groß oder klein schreiben ist übrigens egal, ich werde ihn hier immer groß schreiben. Die kleine Schreibweise ist aber meist praktischer. Zur Korrektur der Befehle bei Tippfehlern stehen nicht allzu viele Möglichkeiten zur Verfügung. Wie im CLI kann nur der jeweils letzte Buchstabe der Zeile mit der [Backspace]-Taste gelöscht werden. Sind Sie mit der Eingabe eines Befehls fertig, so beenden Sie ihn durch Druck auf die [Return]-Taste oder die [Esc]-Taste. Bei Verwendung der [Return]-Taste kehrt der Cursor nach der Ausführung des Befehls zurück in das Text-Fenster. Tippen Sie aber auf die [Esc]-Taste, so bleibt der Cursor auch nach der Ausführung des Befehls in der Kommandozeile und Sie können den nächsten Befehl erteilen.

Die Parameter von Escape-Befehlen sind Texte, die, im Gegensatz zu den Texten für CLI-Befehle, **immer** durch besondere Zeichen begrenzt werden müssen. Im CLI gab es zwar als Begrenzer auch die Anführungszeichen ("); sie waren aber nur bei Texten nötig, die Leerzeichen enthalten. Die Parameter von Escape-Befehlen können durch beliebige Sonderzeichen begrenzt werden, solange sie mit demselben Sonderzeichen aufhören, mit dem sie auch begonnen wurden.

```
"Ein Text" 'Text' /noch einer/ ?Ein 4. Text? ¿komisch, nicht?¿
```

Diese vier Texte sind also vier völlig legitime Parameter für Escape-Befehle mit sehr verschiedenen Begrenzern. Auf diese Weise können auch Texte eingegeben werden, die selbst ein beliebiges Sonderzeichen enthalten.

Bei allen Beschreibungen der verschiedenen Escape-Befehle werden das einleitende Tippen auf die [Esc]-Taste und das beendende [Esc] oder [Return] im allgemeinen nicht erwähnt. Sie sind aber nichtsdestotrotz immer notwendig!

Globale Escape-Befehle

Globale Escape-Befehle sind Befehle, die den Text »als Ganzes« betreffen, ihn also zum Beispiel abspeichern oder Formateinstellungen ändern oder abfragen.

Der erste globale Befehl, den Sie kennengelernt haben, war X, mit dem *ed* verlassen wird, mit dem gleichzeitig aber der Text-Puffer, der die gerade bearbeitete Version des Textes enthält, auf Diskette (beziehungsweise Festplatte) gespeichert wird. Erst ab diesem Moment sind alle Änderungen unwiderruflich gesichert.

Der Befehl Q beendet ebenfalls das Programm, jedoch ohne zuvor die Änderungen auf Diskette beziehungsweise Festplatte zu sichern. Haben Sie aber Änderungen vorgenommen, die noch nicht gesichert wurden, so werden Sie vor dem Verlassen des Programms noch gefragt, ob Sie diese Änderungen wirklich »vergessen« wollen. Nur wenn Sie mit [Y] (für *yes* = *Ja*) antworten, sind die Änderungen wirklich verloren. Alle anderen Tastendrücke brechen den Befehl ab. Sie erhalten so noch eine letzte Chance. Normalerweise werden Sie *ed* aber wohl immer mit [Esc][X] verlassen.

Zum »Zwischendurch-Sichern« Ihres Textes dient der Befehl SA (*save all* oder zu deutsch *sichere alles*). Sie sollten diesen Befehl bei größeren Texten öfter einmal – nach jeder größeren Änderung – anwenden, damit Ihre Arbeit nicht bei jeder mittleren Katastrophe verlorengeht. Schließlich könnte es ja einen Stromausfall geben oder Ihr Sprößling kommt vielleicht auf den Gedanken, Ihrem Computer einmal den Stecker herauszuziehen.

Sollten Sie einmal bemerken, daß die Änderungen, die Sie in einer Zeile gemacht haben, Ihnen nicht mehr gefallen, so können Sie den Befehl [Esc][U] verwenden. U steht dabei für *Undo* (engl. für *rückgängig machen/widerrufen*). Immer wenn Sie in eine neue Zeile kommen und dort Veränderungen beginnen, legt *ed* zuvor noch eine Kopie der alten Zeile in einen Zwischenspeicher. Diese Kopie wird durch den [Esc]-[U]-Befehl wieder »zurückgeholt«.

Text-Blocks

Wenn Sie größere Bereiche in einem Text löschen oder eine Gruppe von Zeilen duplizieren oder verschieben wollen, können Sie *Text-Blocks* verwenden. Es gibt immer genau einen »aktuellen Text-Block«, den Sie durch spezielle Escape-Befehle markieren können. Text-Blocks können gelöscht, an einer anderen Stelle in den Text eingefügt und einzeln auf Diskette gesichert werden.

Den Anfang eines Text-Blocks markieren Sie, indem Sie die entsprechende Zeile mit dem Cursor aufsuchen und dann den Befehl [Esc][B][S] (***B**lock **S**tart*) geben. Gehen Sie dann in die letzte Zeile, die gerade noch zum Block gehören soll, und tippen Sie [Esc][B][E] (***B**lock **E**nde*) ein. Blocks enthalten immer nur ganze Zeilen!

Um den so markierten Block an anderer Stelle einzufügen, gehen Sie mit dem Cursor an diese Stelle (meist an den Beginn einer Zeile) und geben den Befehl [Esc][I][B] (***I**nsert **B**lock*; *Block einfügen*) ein. [I][B] können Sie beliebig oft aufrufen und so beliebig viele Kopien des markierten Blocks an vielen Stellen eines langen Textes einfügen. Wollen Sie danach den Original-Block aus dem Text entfernen, so brauchen Sie nur [Esc][D][B] (***D**elete **B**lock*; *Block löschen*) einzugeben. [D][B] löscht immer den zuletzt markierten Block. Dieser ist danach unwiederbringlich verloren!

Manchmal ist es sinnvoll, einen Text in mehrere kleine Texte aufzuspalten. Dies können Sie erreichen, indem Sie den markierten Block in einer separaten Datei abspeichern (und ihn dann vielleicht löschen). Der Befehl [Esc][W][B] <dateiname> (***W**rite **B**lock*; *Block schreiben*) leistet genau das. <dateiname> muß dabei natürlich durch den Namen einer Datei (in Begrenzer-Zeichen, wie oben beschrieben) ersetzt werden. Auf ähnliche Weise können Sie eine Datei an der aktuellen Cursor-Position in den gerade bearbeiteten Text einfügen, indem Sie [Esc][I][F] <dateiname> (***I**nsert **F**ile*; *Datei einlesen*) eingeben. Die folgende Beispiel-Befehlsfolge ersetzt zum Beispiel den gerade markierten Block durch den Inhalt einer Datei *Neu* aus dem Directory *texte*, wobei zuvor aber der alte Blockinhalt in der Datei *Alt* gesichert wird.

```
[Esc] WB !:texte/Alt!
[Esc] DB
[Esc] IF !:texte/Neu!
```

Befehle zur Bewegung der Schreibmarke

Alle Befehle zur Bewegung der Schreibmarke gibt es auch noch einmal als Escape-Befehle. Ich gehe nicht mehr zu ausführlich darauf ein, da sie ja im wesentlichen bereits bekannt sind.

Der Befehl [Esc] B setzt die Schreibmarke in die letzte Spalte der letzten Zeile des gesamten Textes (nicht des aktuellen Fensters, obwohl dies gelegentlich übereinstimmen kann). Der Befehl [Esc] T setzt die Schreibmarke entsprechend in die erste Spalte der ersten Zeile des gesamten Textes.

Der Befehl [Esc] CE setzt die Schreibmarke in die letzte Spalte der aktuellen Zeile. [Esc] CL bewegt ihn um eine Position nach links. [Esc] CR bewegt die Schreibmarke um eine Position nach rechts und [Esc] CS bewegt sie zum Start der aktuellen Zeile.

Der Befehl [Esc] N setzt die Schreibmarke in die nächste Zeile. [Esc] P setzt ihn in die vorangehende Zeile. Mit dem Befehl [Esc] M und einer Zahl dahinter kann man zu einer bestimmten Zeile in einem langen Text springen. Deren Nummer muß nur bekannt sein. Der folgende Befehl springt so zum Beispiel in die 113te Zeile eines Textes.

```
[Esc] M 113
```

Dies ist besonders für Programmierer interessant, denen der Compiler oder Interpreter oft Programmfehler zusammen mit Zeilenzahlen angibt. Die restlichen Escape-Befehle zur Bewegung der Schreibmarke sind aber wohl nur in Befehlsfolgen interessant (siehe unten), da es wesentlich knappere direkte Befehle für die meisten dieser Bewegungen gibt (siehe oben).

Suchen und Ändern von Text

Während die meisten Escape-Befehle, die Sie bis jetzt kennenlernten, auch mit direkten Befehlen zu erreichen waren, machen einem die Befehle zum Suchen und vor allem zum Ändern von Textstellen das Leben wirklich leichter. Diese Möglichkeiten werden oft – allerdings fälschlicherweise – für den Inbegriff der Textverarbeitung überhaupt gehalten.

Der Befehl [Esc][F] <text> sucht (findet) <text>, ausgehend von der aktuellen Position der Schreibmarke und positioniert die Schreibmarke auf den Beginn des ersten Vorkommens. Falls nötig, wird das Text-Fenster dazu entsprechend verschoben. Genauso verhält sich der Befehl [Esc] BF (*Backwards Find*; *finde rückwärts*), der jedoch nicht von der aktuellen Position nach »unten«, sondern zurück, in Richtung Anfang der Datei, sucht.

[Esc] E /alt/neu/ (*Exchange = austauschen*) sucht den Text */alt/* und ersetzt ihn durch */neu/*, falls er gefunden wird. Wollen Sie sichergehen, daß nur die Textstelle geändert wird, die Sie gemeint hatten, und nicht eine, die zufällig genauso lautet, so können Sie statt E auch EQ (*Exchange and Query = austauschen und nachfragen*) verwenden. Bevor die Textstelle wirklich ersetzt wird, fragt *ed* Sie dann um Bestätigung. Nur wenn Sie [Y] (für *yes = Ja*) antworten, wird die Stelle wirklich geändert. Die folgende Befehlszeile ersetzt zum Beispiel das nächste Vorkommen von *Haus* durch *Hütte* – aber nur, wenn Sie die Ersetzung ausdrücklich bestätigen. (Mit [Ctrl]+[G]) können Sie diesen Befehl danach beliebig oft wiederholen.)

[Esc] EQ /Haus/Hütte/

Achten Sie bei diesem Beispiel-Befehl darauf, daß bei E und EQ die beiden Textstellen durch nur ein Begrenzerzeichen voneinander getrennt werden.

Normalerweise ist es F, E und EQ nicht egal, wie der Text geschrieben ist, der gesucht wird. Ob im Text wirklich *Haus*, *hAUS*, *haUs* und so weiter steht, spielt also wirklich eine Rolle. Wollen Sie aber, daß diese Befehle nicht auf die genaue Schreibweise achten (Groß-/Klein-Schreibung also ignorieren) können Sie den Befehl [Esc] UC (*Upper Case*; *Großschreibung*)

verwenden. Wollen Sie später Kleinschreibung wieder von Großschreibung unterscheiden, so tippen Sie [Esc] LC (*Lower Case*; *Kleinschreibung*) ein.

Einfügen und Löschen von Text

Genauso wie die Befehle zur Cursor-Steuerung, bieten die Befehle zum Einfügen von Text mittels Escape-Befehlen nichts wirklich Neues. Text einzufügen ist mit direkten Befehlen wesentlich leichter. Wenn Sie jedoch erst einmal Ihre ersten Befehlsfolgen mit Wiederholungsfaktoren erstellen, werden Sie die Befehle auch aus dieser Gruppe zu schätzen wissen. Mehr zu diesem Thema aber im folgenden Abschnitt.

Der Befehl [Esc] I <text> (engl. *to **insert** = einfügen*) setzt <text> in eine neue Zeile **vor** die aktuelle (in der sich der Cursor befindet). Genauso setzt [Esc] A <text>(engl. to ***append*** = *anhängen*) [text] in eine neue Zeile **hinter** die aktuelle. Wie üblich muß der einzufügende Text in Begrenzer-Zeichen eingeschlossen werden.

Auch das Zerteilen und Zusammenfügen von Text-Zeilen kann mit Escape-Befehlen genauso wie mit direkten Befehlen erreicht werden. [Esc] S (engl. *to split =teilen*) beendet die Zeile an der aktuellen Position der Schreibmarke und fügt den Rest der Zeile in eine neue Zeile dahinter ein. [Esc] J (engl. *to join = zusammenfügen*) fügt die Folgezeile an das Ende der aktuellen an. Die Befehlsfolge [Esc] S [Return] [Esc] J [Return] hebt sich also selbst wieder auf und bewirkt gar nichts. Mit [Esc] D (engl. *to **delete** = löschen*) kann die gesamte aktuelle Zeile, mit [Esc] DC nur der aktuelle Buchstabe gelöscht werden. [Esc] DC wirkt also genau so wie das Drücken der [Del]-Taste und [Esc]D wie [Ctrl]+[B].

Einige zeitsparende Tricks

Statt jedesmal nur einen Escape-Befehl einzutippen, können Sie auch gleich mehrere hintereinander eingeben. Mehrere Befehle in einer Zeile werden durch Semikolons (;) getrennt. Die Buchstabenfolge [Esc] T ; N ; N [Return] setzt den Cursor also auf etwas umständliche Weise in die dritte Zeile (der M-Befehl wäre etwas einfacher).

Soll ein und derselbe Befehl mehrfach ausgeführt werden, so kann man dies auf zwei verschiedene Weisen erreichen. Einmal kann man das direkte Kommando [Ctrl]+[G] verwenden, mit dem jeweils der letzte Escape-Befehl wiederholt wird. Besonders bei Such- und Änderungs-Befehlen ist das ganz praktisch, wenn zum Beispiel die erste Fundstelle nicht das Vorkommen des Textes ist, den man eigentlich gesucht hat.

Wenn man schon vorher weiß, wie oft ein Befehl ausgeführt werden soll, schreibt man einfach eine Zahl vor den Befehl. [Esc] 4 E /der/die/ ändert zum Beispiel die nächsten vier Vorkommnisse von *der* in *die* um.

Will man mehrere Befehle zusammen mehrfach ausführen, so kann man diese Befehle zusammen in die Befehlszeile schreiben, mit runden Klammern gruppieren und die gewünschte Anzahl von Wiederholungen vor die Klammer schreiben.

Statt einer Zahl kann man immer auch RP (engl. *to **repeat** = wiederholen*) verwenden. Der Befehl wird dann so lange wiederholt, bis ein Fehler passiert. Ein solcher Fehler wäre zum

Beispiel der Befehl, eine Zeile nach oben zu gehen, wenn der Cursor bereits in der ersten Zeile ist. Der Befehl [Esc] RP N ist also eine etwas umständliche Methode, um in die letzte Zeile einer Datei zu gelangen.

9.2.4 Liste aller ed-Befehle

Direkte Befehle:

Befehl	Wirkung
[Backspace]	Buchstaben links von Schreibmarke löschen
[Del]	Buchstaben an der Schreibmarkenposition löschen
[Return]	Zerteilen der aktuellen Zeile in zwei
[Tab]	Schreibmarke n Schritte nach rechts
[Ctrl]+[A]	Neue Zeile einfügen
[Ctrl]+[B]	Löschen einer Zeile
[Ctrl]+[D]	Text um 12 Zeilen nach unten schieben
[Ctrl]+[E]	Schreibmarke zum Anfang (oder Ende) des Fensters
[Ctrl]+[F]	Groß-/Kleinschreibung ändern
[Ctrl]+[G]	Wiederholen des letzten Escape-Befehls
[Ctrl]+[H]	gleich [Backspace]
[Ctrl]+[H]	gleich [Tab]
[Ctrl]+[O]	Rest des Wortes oder Leerraums löschen
[Ctrl]+[R]	Schreibmarke zum Ende des vorangehenden Wortes
[Ctrl]+[T]	Schreibmarke zum Anfang des nächsten Wortes
[Ctrl]+[U]	Text um 12 Zeilen nach oben schieben
[Ctrl]+[Y]	Rest der Zeile löschen
[Ctrl]+»]«	Schreibmarke zum Anfang (oder Ende) der Zeile
[Ctrl]+»[«	gleich [Esc]

Escape-Befehle:

Befehl	Wirkung
[Esc] A /*Text*/	*Text* hinter aktueller Zeile einfügen
[Esc] B	Schreibmarke zum Ende des Textes
[Esc] BE	Block-Ende bei Schreibmarke markieren
[Esc] BF /*Text*/	*Text* rückwärts suchen
[Esc] BS	Block-Start bei Schreibmarke markieren
[Esc] CE	Schreibmarke zum Ende der Zeile
[Esc] CL	Schreibmarke eine Position nach links
[Esc] CR	Schreibmarke eine Position nach rechts
[Esc] CS	Schreibmarke zum Start der Zeile
[Esc] D	Aktuelle Zeile löschen
[Esc] DB	Block löschen
[Esc] DC	gleich [Del]
[Esc] E *alt*/*neu*/	Ändere nächste Textpassage *alt* in *neu*
[Esc] EQ	Ändere nächste Textpassage *alt* in *neu* ; frage vorher um Bestätigung
[Esc] EX	Rechten Rand ausdehnen
[Esc] F /*Text*/	Suche *Text*
[Esc] I /*Text*/	*Text* vor aktueller Zeile einfügen
[Esc] IB	Block bei Schreibmarke einfügen
[Esc] IF /*Name*/	Datei *Name* bei Schreibmarke einfügen
[Esc] J	Folgende Zeile an das Ende der aktuellen Zeile hängen
[Esc] LC	Groß-/Kleinschreibung (bei F, E und EQ) beachten
[Esc] M *n*	Schreibmarke nach Zeile *n*
[Esc] N	Schreibmarke in die nächste Zeile
[Esc] P	Schreibmarke in die vorangehende Zeile
[Esc] Q	Programm ohne Sicherung des Textes beenden
[Esc] RP	Folgenden Befehl wiederholen, bis ein Fehler auftritt
[Esc] S	gleich [Return]
[Esc] SA	Textpuffer auf Diskette sichern
[Esc] SB	Aktuellen Block im Fenster zeigen
[Esc] SH	Informationen über den aktuellen Text zeigen
[Esc] SL *n*	Linken Rand auf *n* setzen
[Esc] SR *n*	Rechten Rand auf *n* setzen
[Esc] ST *n*	Abstand zwischen zwei Tabstops auf *n* setzen
[Esc] T	Schreibmarke zum Anfang des Textes

[Esc] U	Alle Änderungen an der aktuellen Zeile rückgängig machen
[Esc] UC	Groß-/Kleinschreibung (bei F, E und EQ) nicht beachten
[Esc] WB /*Name*/	Aktuellen Block in Datei *Name* sichern
[Esc] X	Beenden von *ed* mit vorherigem Sichern des aktuellen Textes

9.3 Der mausgesteuerte Editor »MicroEmacs«

Während *ed* ein eher simpler Texteditor ist, wie man ihn oft kostenlos zu einem Computer dazu bekommt, ist *MicroEmacs* ein recht leistungsfähiges Programm, das es durchaus mit den besten kommerziellen Produkten aufnehmen kann. *MicroEmacs* kann sowohl ausschließlich über die Tastatur, als auch (nahezu) ausschließlich über die Maus und Menübefehle gesteuert werden. Er stellt damit einen nahezu idealen Kompromiß für ein Programm dar, das für den Anfänger leicht zu erlernen ist und dem geübten Profi trotzdem schnelles Arbeiten ermöglicht. Obwohl *MicroEmacs* eigentlich als Programmeditor »gedacht« ist, zeigt er seine Vorteile gegenüber dem simplen *ed* auch schon bei relativ kurzen simplen Texten – wie zum Beispiel Kommandofolgen.

Das Programm ist andererseits aber so vielseitig und die Befehlsliste so lang, daß ich es im Rahmen dieses Buches nicht vollständig beschreiben kann. Sie finden in den folgenden Abschnitten deshalb nur die wichtigsten Befehle erläutert. (Im letzten Abschnitt zum Thema *MicroEmacs* finden Sie allerdings einen kompletten Überblick über alle Befehle mit knappen Erläuterungen.) Mit etwas Experimentierfreude werden Sie aber bald auch die restlichen Befehle kennenlernen – bei einem menügesteuerten Programm ist das meist recht einfach. Und wenn Sie einigermaßen solide Englisch-Kenntnisse besitzen, können Sie sich auch das Kurzhandbuch zu *MicroEmacs* ausdrucken, das sich unter dem Namen *MicroEmacs.doc* in derselben Schublade wie auch das Programm selbst befindet.

9.3.1 Die Geschichte von MicroEmacs

MicroEmacs auf dem Amiga hat eine interessante Vorgeschichte, die ich Ihnen nicht vorenthalten möchte. Falls Sie an solchen »Histörchen« nicht interessiert sind, können Sie diesen Abschnitt natürlich übergehen und gleich mit dem folgenden fortfahren.

MicroEmacs (was man manchmal auch *µEmacs* schreibt) ist aus dem berühmten Editor *emacs* hervorgegangen, der zunächst nur auf Großrechnern zu finden war und sich bis heute auf UNIX-Systemen großer Beliebtheit erfreut. Dieser Editor war speziell für die Programmentwicklung geschrieben worden und wird auch meist für das Editieren von Programmtexten eingesetzt. Insbesondere die Spracheigenschaften von C und Lisp wurden und werden von *emacs* besonders gut unterstützt, ähnlich wie der Editor im List-Fenster von Amiga-BASIC besonders gut mit BASIC-Programmen fertig wird.

Als die ersten Mikrocomputer auftauchten und bald weite Verbreitung fanden, waren viele der ersten Benutzer ehemalige Programmierer von Großrechnern. Diese aber waren – besonders an den amerikanischen Universitäten – oft eingeschworene *emacs*-Fans und trauerten ihrem geliebten Editor nach. Dies nicht zuletzt deshalb, weil die entsprechenden Editoren auf

Mikrocomputern sehr primitiv waren. Aus dieser unglücklichen Situation und dem Bemühen, sie zu beseitigen, ging *MicroEmacs* hervor. Eine »abgespeckte« Version des großen *emacs*, die vollständig in der Programmiersprache C geschrieben war und fast alle vertrauten Eigenschaften von *emacs* beinhaltet. Nur die leistungsfähigsten und aufwendigsten Fähigkeiten des großen *emacs* wurden aus Platz- und Zeit-Gründen nicht übernommen.

Da man bei der Programmierung von *MicroEmacs* von vorneherein darauf geachtet hatte, möglichst keine Eigenheiten eines speziellen Computers zu benutzen, tauchten in der Folgezeit für fast alle leistungsfähigen PCs Versionen des *MicroEmacs* auf. Diese wurden meist mit viel Liebe von den Fans des jeweiligen Computers an ihren Rechner angepaßt und umsonst an die Kollegen Programmierer verteilt. Diese Art umsonst verteilter Software wird üblicherweise »Public-Domain-Software« genannt. Auch heute noch kann man den Programmtext von *MicroEmacs* umsonst erhalten und sich mit kleinen Änderungen daran ganz persönliche Versionen dieses Editors für seinen eigenen Computer schaffen.

Auch auf dem Amiga waren die Programmierer von Anfang an mit den zur Verfügung stehenden Werkzeugen (hauptsächlich *ed*) sehr unzufrieden und begannen *MicroEmacs* an den Amiga anzupassen. Die ersten Ergebnisse dieser Bemühungen waren noch recht anfällig und nutzten die Fähigkeiten des Amiga so gut wie überhaupt nicht. Mit der Zeit wurden dann aber Pull-down-Menüs für alle wichtigen Befehle eingeführt und auch die Maus zur Cursorsteuerung eingesetzt. So entstand die heutige Version von *MicroEmacs*. Man kann sie immer noch (inklusive Programmtext) umsonst erhalten und dann damit machen, was man will.

Da man auch bei Commodore mit *ed* als Programmeditor recht unzufrieden war, entschloß man sich, die damals verfügbare *MicroEmacs*-Version noch etwas aufzupolieren, Fehler zu entfernen und sie schließlich auf der Amiga-Toolkit-Diskette allen interessierten Amiga-Besitzern zur Verfügung zu stellen. (Im Augenblick wird sie noch auf der Extras-Diskette jedem Amiga 500 beigelegt. Das muß aber nicht so bleiben.) Und so kommen Sie als Anwender des Amiga 500 heute – dank der Bemühungen der internationalen Programmierer-Gemeinschaft – zu einem Texteditor, der den guten alten *ed* um Längen schlägt und auch für die professionelle Programmentwicklung durchaus geeignet ist!

Nach dieser etwas ausschweifenden Vorrede werden Sie nun aber auch wirklich erfahren, was es mit *MicroEmacs* auf sich hat.

9.3.2 Wie Sie MicroEmacs aufrufen und verlassen

Wie bei jedem anderen Programm auch, sind die wichtigsten Informationen zu *MicroEmacs*, wie man hinein- und wieder hinauskommt. Oder anders: wie man es startet und beendet. Bei *MicroEmacs* haben Sie hierzu mehrere Möglichkeiten. Welche Sie nutzen, hängt davon ab, ob Sie sich gerade auf der Workbench oder im CLI befinden und ob Sie gerne mit der Maus oder lieber mit Tastaturbefehlen arbeiten.

MicroEmacs befindet sich in der Schublade *Tools* (der augenblicklichen Version) der Extras-Diskette. Sie müssen diese Diskette nur in ein Laufwerk Ihres Amiga 500 einlegen, das

Disketten-Piktogramm öffnen, das Schubladen-Piktogramm *Tools* öffnen und können *MicroEmacs* dann mit einem Doppelklick auf sein Piktogramm starten.

Im CLI können Sie mit dem *path*-Befehl *MicroEmacs* »auffindbar machen« und dann starten, indem Sie den Programmnamen eingeben:

```
1>path ExtrasD:Tools ADD
1>microemacs
```

Wenn Sie nur ein Laufwerk besitzen, müssen Sie dazu unter Umständen allerdings zwei Diskettenwechsel durchführen. Denn für den *path*-Befehl werden die Workbench- und die Extras-Diskette und für den Aufruf von *MicroEmacs* die Extras-Diskette benötigt.

Wenn Sie vorhaben, *MicroEmacs* häufiger zu verwenden, ist es wahrscheinlich sinnvoll, ihn auf die Workbench-Diskette zu legen (und *ed* sowie *edit* zu löschen). Legen Sie ihn am besten (auf der Workbench) in die Schublade *Utilities*, in der auch *Notepad* liegt. Dieses Directory (Schublade) liegt normalerweise schon im Suchpfad. Sie können *MicroEmacs* dann vom CLI und von der Workbench aus relativ leicht starten und können sich den soeben beschriebenen *path*-Befehl sparen.

Als CLI-Benutzer ist Ihnen der Name *MicroEmacs* wahrscheinlich zu lang und fehleranfällig. Sie können *MicroEmacs* aber natürlich jederzeit umbenennen. Tun Sie das am besten auf der Workbench. Ein sinnvoller und kürzerer Name wäre zum Beispiel *emacs*.

Egal, auf welchen (Um-)Wegen Sie *MicroEmacs* starten, erscheint kurz darauf der Bildschirm von *MicroEmacs*. Das Programm öffnet einen eigenen Screen ohne Fenster darin. Dieser Screen ist blau wie der Workbench-Screen und die Schrift darin erscheint in Weiß – wenn Sie die Workbench-Farben nicht mit *Preferences* geändert haben. Er enthält eine Titelleiste am oberen Rand (die auch diverse Menüs enthält) und die üblichen Front- und Back-Gadgets, mit denen Sie den Screen hinter den Workbench-Screen legen und wieder hervorholen können.

Der Screen von *MicroEmacs* enthält, wie schon erwähnt, keine Fenster. Alles spielt sich auf der gesamten Fläche des Bildschirms ab. Diese Fläche wird durch eine Linie geteilt, die das untere Viertel optisch abtrennt. Diese Linie ist gestrichelt und enthält am linken Rand einige Informationen (zum Beispiel den Namen der gerade bearbeiteten Datei – sobald Sie eine einlesen). Oberhalb der Linie steht der jeweils editierte Text. Unterhalb der Linie erscheinen Nachrichten und Fehlermeldungen. Weiterhin müssen Sie hier auch Texte eingeben, wenn dies für die Ausführung eines Befehls nötig ist. Dieser Bildschirmbereich hat ähnliche Aufgaben wie die Kommandozeile von *ed* und heißt auch hier »Kommandozeile« oder »Befehlszeile«.

```
MicroEMACS V1.1a
echo "A500 Workbench 1.2 D  Version 33.56  23-APR-87*N"
BindDrivers
if EXISTS sys:system
   path sys:system add
endif
if EXISTS sys:utilities
   path sys:utilities add
endif
Dir RAM:
Path RAM: add
SetMap d
Addbuffers df0: 20 ;this uses up about 10K of memory, but improves disk speed
LoadWb
failat 30
SetClock >NIL: Opt load
Date
endcli > nil:


-- MicroEMACS -- STARTUP-SEQUENC -- File: STARTUP-SEQUENCE ------------------
[Read 17 lines]
```

Bild 9.3: *MicroEmacs mit der Datei Startup-Sequence in Bearbeitung*

Betrachten Sie sich nun einmal die Menüs von *MicroEmacs*, indem Sie bei gedrückter Menütaste mit dem Mauszeiger in der Titelleiste hin- und herfahren – aber bitte erst einmal, ohne einen Befehl daraus auszuwählen. Es sind sieben Menüs, die die gesamte Titelleiste einnehmen. Nahezu jeder dieser Menübefehle hat ein Tastaturäquivalent, das rechts neben dem Namen des Befehls aufgelistet wird. Es sind aber keine üblichen Tastaturäquivalente, für die Sie die Commodore- oder die Amiga-Tasten festhalten müssen. Statt dessen werden alle Tastaturäquivalente durch Drücken der [Ctrl]-Taste, die in den Menüs durch den Dachakzent (^) symbolisiert wird, oder der [Esc]-Taste, die in den Menüs durch *ESC* bezeichnet ist, eingeleitet. Zudem bestehen diese Tastaturäquivalente manchmal nicht nur aus einer Tastenkombination, sondern aus mehreren Tasten, die Sie nacheinander drücken müssen. Das Tastaturäquivalent *ESC^V* bedeutet zum Beispiel, daß Sie zunächst die [Esc]-Taste drücken müssen und dann die Tastenkombination [Ctrl]+[V]. Und ^X^S bedeutet, daß Sie nacheinander die beiden Tastenkombinationen [Ctrl]+[X] und [Ctrl]+[S] drücken müssen.

Auch der Befehl zum Beenden von *MicroEmacs* hat ein solches Tastaturäquivalent. Wenn Sie *MicroEmacs* verlassen wollen, müssen Sie entweder den Befehl *Quit* aus dem Menü *Project* wählen oder die Tastenkombination [Ctrl]+[C] (^C) drücken – die nicht zufällig auch sonst im CLI dafür eingesetzt wird, Programme abzubrechen. *MicroEmacs* wird aber nicht unbedingt sofort beendet, wenn Sie [Ctrl]+[C] drücken. Wenn Sie einen Text eingegeben oder geändert haben und wollen das Programm verlassen, ohne den Text vorher gesichert zu haben, erscheint die Frage *Modified buffers exist, do you really want to exit [y/n]?*. Nur wenn Sie daraufhin die »Y« oder »y« eingeben und die [Return]-Taste drücken, wird das Programm

wirklich verlassen. Mit der Eingabe »N« oder »n« bleiben Sie im Programm und können die Änderungen erst sichern.

9.3.3 Wie Sie Texte abspeichern und einlesen

Das Einlesen und Abspeichern eines Textes läuft mit *MicroEmacs* im wesentlichen genau so ab wie im *ed*. Wählen Sie zum Einlesen eines Textes von Diskette den Befehl *Open (Read)* aus dem Menü *Project*. Unten am Bildschirm erscheint daraufhin die Meldung *Read file* und dahinter ein Doppelpunkt. Geben Sie hinter diesem Doppelpunkt den Pfadnamen einer Textdatei ein und drücken Sie auf [Return]. *MicroEmacs* liest diese Datei dann in seinen »Puffer« (engl. *buffer*) ein und präsentiert sie Ihnen im oberen Bereich des Bildschirms. In diesem Puffer (im RAM-Speicher) werden zunächst alle Änderungen durchgeführt. Gibt es noch keine Datei dieses Namens, wird eine neue angelegt. Zunächst existiert diese Datei aber erst im Puffer. Erst wenn Sie diesen Puffer sichern, wird auch auf der Diskette oder Festplatte eine neue Datei erzeugt.

Noch einfacher als das Einlesen ist das Abspeichern eines neuen oder geänderten Textes. Wählen Sie dazu einfach den Befehl *Save* aus dem Menü *Project*. *MicroEmacs* speichert dann den aktuellen Inhalt des Puffers in die Datei, deren Name in der Trennlinie zwischen oberem und unterem Bildschirmbereich zu lesen ist. Sie können den Inhalt eines Puffers auch unter anderem Namen speichern. Wählen Sie hierzu den Befehl *Save As* aus dem Menü *Project* und *MicroEmacs* fragt Sie nach dem Namen der Datei, in die der Puffer geschrieben werden soll. Und schließlich hat der Save-Befehl auch noch eine dritte Variante. Wählen Sie *Save/exit* aus dem Menü *Project*, wird *MicroEmacs* wie bei *Quit* verlassen. Vorher wird aber noch der aktuelle Pufferinhalt, wie mit dem Befehl *Save*, gesichert.

9.3.4 Wie Sie sich im Text bewegen

Genau wie im *ed*, dem anderen Editor des Amiga, gibt es auch hier wieder eine Schreibmarke, »dot« (engl. für *Punkt*) genannt, die die Stelle markiert, an der neu eingegebener Text am Bildschirm erscheint. Diese Schreibmarke können Sie auf viele verschiedene Arten bewegen; einige davon kennen Sie bereits aus anderen Programmen.

Die Cursortasten bewegen die Schreibmarke zum Beispiel in die auf der jeweiligen Taste angegebene Richtung. Halten Sie die Shift-Taste fest, bevor Sie eine Cursor-Taste drücken, springt die Schreibmarke bis zum »Anschlag« in der jeweiligen Richtung, also bis zum oberen, unteren, rechten oder linken Rand des Bildschirms. Mit der Maus können Sie die Schreibmarke ebenfalls bewegen. Sie brauchen – wie im *Notepad* – einen Buchstaben nur anzuklicken und schon springt die Schreibmarke an diese Stelle.

Darüber hinaus kennt *MicroEmacs* aber noch eine Vielzahl anderer Möglichkeiten zur Bewegung der Schreibmarke und zur Verschiebung des am Bildschirm sichtbaren Textausschnitts. Zu jeder dieser Bewegungsmöglichkeiten gibt es einen Menübefehl im Menü *Move* (engl. *to move = bewegen*) – obwohl gerade für diesen Anwendungsfall die Tastaturäquivalente dieser Befehle wohl »handlicher« sind. Ich werde der Lesbarkeit wegen in den folgenden Beschreibungen aber die Befehlsnamen verwenden.

Wenn Sie schnell zum Anfang oder Ende des gesamten Textes wollen, so können Sie das mit den beiden Befehlen *Top of buffer* beziehungsweise *End of buffer*. Die Befehle *Top of window* und *End of window* setzen die Schreibmarke in entsprechender Weise an den oberen beziehungsweise unteren Bildschirmrand. Mit *Goto Line* können Sie die Schreibmarke in eine bestimmte Zeile des Textes setzen, deren laufende Nummer Sie dazu allerdings kennen müssen. Diese Nummer wird am unteren Bildschirmrand abgefragt, wenn Sie diesen Befehl aufrufen.

Besonders interessant ist der Befehl *Swap dot&mark*, mit dem Sie jederzeit zu einer bestimmten Stelle im Text springen können, die Sie einmal markiert haben. Zum Markieren dieser Stelle dient der Befehl *Set Mark* im *Edit*-Menü, der im übernächsten Abschnitt erläutert wird. *Swap dot&mark* tauscht die Schreibmarke und diese Markierung aus. Das heißt, nach dem Befehl ist die Schreibmarke dort, wo die Markierung war und als markiert gilt der Buchstabe, auf dem zuvor die Schreibmarke stand.

Die restlichen Befehle im *Move*-Menü sind dann wieder leichter verständlich. *Next page* und *Prev page* bringen die nächste beziehungsweise vorangehende Seite des Textes auf den Bildschirm. Als »Seite« gilt dabei die Höhe des oberen Bildschirmbereichs, in dem der Text angezeigt wird. Mit den beiden Befehlen *Next Word* und *Previous Word* kann man sich innerhalb einer Zeile rasch bewegen. Diese Befehle setzen die Schreibmarke auf den Buchstaben vor dem nächsten Wort beziehungsweise den ersten Buchstaben des vorangehenden Wortes (ausgehend von der augenblicklichen Position der Schreibmarke). Und *Scroll Up* und *Scroll Down* schließlich verschieben nur den am Bildschirm sichtbaren Ausschnitt des Textes eine Zeile nach oben beziehungsweise nach unten – wobei die Schreibmarke aber an derselben Stelle im Text stehenbleibt.

9.3.5 Wie Sie Text eingeben und bearbeiten

Wie schon erwähnt, erscheint Text, den Sie (über die Tastatur) neu eingeben, wie in allen anderen Amiga-Editoren vor der Schreibmarke und wird in den Text eingefügt (überschreibt also nicht vorhandenen Text). Insofern dürfte Ihnen *MicroEmacs* schon vertraut erscheinen. Aber auch die meisten »besonderen« Tasten wirken wie bei *ed* oder wie im *Notepad*.

[Return] beginnt zum Beispiel eine neue Zeile oder bricht eine alte Zeile in zwei Zeilen auf, wenn die Schreibmarke zuvor mitten in einer Zeile gestanden hat.

Mit [Tab] fügen Sie vor der Schreibmarke einen Tabulator in den Text ein. Dieser Tabulator ist im *MicroEmacs* ein besonderes Zeichen, das »so aussieht wie« mehrere Leerstellen, aber nur ein Zeichen ist. Zu wievielen Leerstellen *MicroEmacs* ein Tabulatorzeichen ausdehnt, können Sie mit einem speziellen Befehl einstellen – nach dem Start sind es acht Leerstellen.

Mit [Del] und [Backspace] können Sie wie üblich kleine Mengen Text löschen. [Del] löscht den Buchstaben »an« der Schreibmarke, [Backspace] den davor. Zum Löschen größerer Textmengen kennt *MicroEmacs* ähnlich wie *Notepad* und *ed* das Konzept des »aktuellen Textblocks«.

9.3.6 Wie Sie mit Textblöcken arbeiten

In *MicroEmacs* werden Anfang und Ende des aktuellen Textblocks durch zwei Positionen bestimmt, die in den Menüs *dot* (engl. für *Punkt*) und *mark* (engl. für *Markierung/Marke*) genannt werden. Der *dot* entspricht dabei der aktuellen Position der Schreibmarke, während die Marke einmal festgelegt wird und dann stehenbleibt, auch wenn sich die Schreibmarke bewegt. Um die Marke festzusetzen, müssen Sie zunächst die Schreibmarke an die gewünschte Position bringen und dann den Befehl *Set Mark* aus dem Menü *Edit* auswählen. Der in diesem Moment hervorgehobene Buchstabe ist dann *mark*. Einfacher ist es – und bewirkt dasselbe – wenn Sie den Buchstaben, auf dem die Marke stehen soll, zweimal anklicken (also einen »Doppelklick« darauf machen).

Mit dem Befehl *Kill region* können Sie den von *dot* und Marke begrenzten Bereich dann löschen. Alle Buchstaben zwischen der Marke und der aktuellen Position der Schreibmarke (dem *dot*), einschließlich der beiden Buchstaben, an denen sich Schreibmarke und Marke befinden, verschwinden dann aus dem Text. Sie landen im sogenannten *kill buffer* (Lösch-Puffer), einer Zwischenablage, aus der Sie den gelöschten Text jederzeit wieder herausholen können.

Wählen Sie dazu den Befehl *Yank* aus dem Menü *Edit*, und der Inhalt des Lösch-Puffers wird vor der Schreibmarke in den Text eingefügt. Wenn Sie also kurz hintereinander *Kill region* und dann *Yank* aufrufen, so wirkt das, als hätten Sie überhaupt nichts getan – die beiden Befehle heben sich auf. Mit *Kill region* und *Yank* können Sie natürlich auch leicht große Textpassagen innerhalb eines Textes verschieben. Dazu löschen Sie die Textpassage, gehen dann mit der Schreibmarke an die gewünschte neue Position und geben den Befehl *Yank ein*. *Yank* können Sie natürlich auch mehrfach hintereinander aufrufen und so einen Text vervielfältigen.

9.3.7 Wie Sie Texte suchen und ersetzen

Wie jeder gute Editor, besitzt *MicroEmacs* natürlich auch eine leistungsfähige Suche&Ersetze-Funktion. Mit *Search forw* und *Search back* können Sie eine kurze Textpassage, ausgehend von der aktuellen Position der Schreibmarke vorwärts oder rückwärts suchen lassen. Dazu erscheint, nachdem Sie einen dieser beiden Befehle ausgewählt haben, die Meldung *Search* am unteren Bildschirmrand und dahinter ein Doppelpunkt. Wenn Sie dann hinter dem Doppelpunkt den Suchbegriff eingeben und mit [Return] abschließen, beginnt die Suche. Findet *MicroEmacs* den Suchbegriff, wird er am Bildschirm gezeigt und die Schreibmarke befindet sich auf dem ersten Buchstaben **hinter** dem Suchbegriff. Wenn Sie *Search forw* oder *Search back* später ein zweites Mal auswählen, erscheint der Suchbegriff, den Sie beim ersten Mal gesucht haben, in eckigen Klammern hinter der Meldung *Search* (also zum Beispiel *Search [Markus]:*). Dann brauchen Sie nur auf [Return] zu drücken, um noch einmal diesen Begiff zu suchen. Wenn Sie aber einen anderen Suchbegriff eingeben und dann [Return] drücken, wird der neue Suchbegriff gesucht.

Im Gegensatz zu den anderen Editoren des Amiga reagieren die Suchfunktionen des *MicroEmacs* auf Unterschiede in der Groß-/Kleinschreibung. Wenn Sie also *Markus* suchen, würde *markus* nicht gefunden. In der beiliegenden Dokumentation wird zwar das Gegenteil

beschrieben (*MicroEmacs* wäre nicht *case sensitive*), bei der mir vorliegenden *MicroEmacs*-Version reagierten alle Suchfunktionen auf Unterschiede in der Groß-/Kleinschreibung, das heißt, man muß jeden Suchbegriff genau so schreiben, wie er auch im Text auftaucht.

Dasselbe gilt auch für die eigentliche Suche&Ersetze-Funktion. Wollen Sie eine kurze Textpassage, die häufiger in einem Text vorkommt, gegen eine andere ersetzen, so müssen Sie diese zunächst einmal suchen. Wählen Sie dazu den Befehl *Search/rep.* aus dem Menü *Search*. Daraufhin erscheint wieder die Aufforderung einen Suchbegriff einzugeben (Search:) am unteren Bildschirmrand. Geben Sie dann das Wort oder die Textpassage ein, die Sie ersetzen wollen und drücken Sie [Return]. Sobald *MicroEmacs* den Suchbegriff das erste Mal gefunden hat, erscheint die Meldung *Replace:* am unteren Bildschirmrand. Tragen Sie dann hinter dem Doppelpunkt den Text ein, der den Suchbegriff ersetzen soll und drücken Sie wieder [Return]. *MicroEmacs* ersetzt dann alle Vorkommnisse des Suchbegriffs hinter der Position der Schreibmarke gegen den hinter *Replace:* eingetragenen Text. Wenn dieser Vorgang abgeschlossen ist (er kann bei großen Textdateien eine Minute oder länger dauern), meldet *MicroEmacs,* wieviele Vorkommnisse des Suchbegriffs ersetzt wurden: *Replaced <nn> occurrences*, wobei statt <nn> natürlich die Anzahl der durchgeführten Ersetzungen auftaucht.

Wenn Sie nicht alle, sondern nur einige Vorkommnisse eines Suchbegriffs durch einen anderen Text ersetzen wollen, können Sie den Befehl *Query s&r* im *Search*-Menü verwenden. Er verhält sich exakt so wie der *Search/rep.*-Befehl – nur wird vor jedem Ersetzen des Suchbegriffs dieser in der oberen Bildschirmhälfte gezeigt und dann gefragt *Change string [y/n/c]?*. Auf diese Frage müssen Sie mit einem Tastendruck antworten ([Return] ist nicht nötig!). Antworten Sie mit [Y], wird der Suchbegriff ersetzt und die Suche fortgesetzt. Antworten Sie mit [N], wird der oben angezeigte Suchbegriff nicht ersetzt, die Suche aber fortgesetzt. Und wenn Sie mit [C] antworten, werden alle restlichen Vorkommnisse des Suchbegriffs bis zum Ende der Datei ohne weitere Rückfragen ersetzt. Wenn Sie statt dessen aber die Suche abbrechen wollen, können Sie das mit der Tastenkombination [Ctrl]+[G].

9.3.8 Wie Sie mit mehreren Puffern und Fenstern arbeiten

Für den intensiven Anwender – speziell den Programmierer – bietet *MicroEmacs* noch einen besonderen Leckerbissen. Dieser verbirgt sich hinter dem häufigen Auftreten des Wortes *buffer* oder *buffers* in den Menüs in Meldungen des Programms. Dieses Wort bezieht sich ja auf den Puffer im RAM-Speicher, in dem jeder Text zunächst bearbeitet wird, bevor er eventuell (!) auf Diskette gespeichert wird. Von diesem Puffer kann es nicht nur ein Exemplar, sondern gleich mehrere geben. Sie können also auch zwei oder drei Texte (mehr oder weniger) gleichzeitig bearbeiten.

Hierzu müssen Sie den zweiten (und alle folgenden) Text(e) nur mit dem Befehl *Visit file* im Menü *Project* und nicht mit dem Befehl *Open* einlesen. Mit *Visit file* wird der alte Pufferinhalt (also die zuerst eingelesene Datei) nicht zerstört und durch die neue Datei ersetzt, sondern statt dessen die neue Datei in den zweiten Puffer eingelesen. Nach *Visit file* ist dann zunächst die neue Datei am Bildschirm zu sehen und die alte verschwindet. Sie können aber jederzeit die Datei, die Sie sehen und bearbeiten wollen ändern, indem Sie den Befehl *Select buffer* im Menü *Edit* aufrufen und dann den Namen der Datei eingeben, den Sie sehen wollen. Eine Liste mit

den Namen und Größen aller schon geladenen Dateien können Sie sich mit *List buffers* im Menü *Edit* anzeigen lassen. In dieser Liste wird auch (durch einen Stern in der ersten Spalte) markiert, welche Puffer modifiziert (und noch nicht abgespeichert) wurden und welche nicht.

Auf diese Art Zugriff auf mehrere Texte in verschiedenen Dateien zu haben, ist besonders dann praktisch, wenn man Textpassagen aus einer in eine andere Datei kopieren möchte. Legen Sie die zu kopierende Textpassage einfach mit *Kill region* in den Lösch-Puffer und fügen Sie sie mit *Yank* dann gleich wieder in den Text ein. Eine Kopie des zuvor gelöschten Textes liegt dann aber immer noch im Lösch-Puffer. Nun wählen Sie mit *Select buffer* die zweite Datei (in die Sie den Text einfügen wollen), setzen die Schreibmarke an die gewünschte Stelle und können dann mit einem erneuten *Yank* eine Kopie der Textpassage aus der anderen Datei einfügen.

Die bislang beschriebene Handhabung mehrerer Puffer ist aber noch recht umständlich. Bei jedem Wechsel, von einem Puffer zum anderen, muß man zum Beispiel den Namen der jeweiligen Datei angeben. Praktisch wäre es auch, wenn man mehrere Puffer auf einmal sehen könnte. Genau das ist auch möglich. Mit dem Befehl *Split window* im Menü *Window* können Sie den oberen Teil des Bildschirms, der den bearbeiteten Text zeigt, in zwei Hälften teilen. Die Dokumentation von *MicroEmacs* nennt diese Hälften Windows; sie haben aber recht wenig mit den komfortablen Fenstern von Intuition zu tun, die Sie aus der Workbench und aus anderen Programmen her kennen. Sie können zum Beispiel nicht verschoben werden, sind immer so breit wie der ganze Bildschirm und können sich nicht überlappen. Trotzdem bieten auch diese simplen Fenster aber die wertvolle Möglichkeit, zwei oder mehr Texte gleichzeitig am Bildschirm betrachten zu können.

Nach dem Teilen des ersten Fensters (das ja fast den ganzen Bildschirm einnahm) enthalten beide Fenster zunächst einmal denselben Text. Sie können mit den Befehlen *Next Window* und *Prev Window* zwischen diesen beiden Fenstern hin- und herspringen und – indem Sie zum Beispiel die Cursortasten benutzen – sich unterschiedliche Teile dieses einen Textes zeigen lassen. Sie könnten sich zum Beispiel im oberen Fenster den Anfang und im unteren das Ende einer langen Textdatei zeigen lassen. Das kann manchmal ganz praktisch sein.

Immer eines der Fenster (es können auch mehr als zwei sein, wenn Sie wiederholt *Split window* aufrufen) ist das aktuelle. Sie erkennen es daran, daß sich in ihm die Schreibmarke befindet. Wenn Sie nun die Befehle *Open, Visit file* oder *Select buffer* aufrufen, so taucht die dadurch auf den Bildschirm kommende Datei im aktuellen Fenster auf. Wenn Sie also zwei verschiedene Texte gleichzeitig sehen wollen, müssen Sie zunächst das eine Fenster teilen (*Split window*) und dann mit *Open* den zweiten Text einlesen oder ihn mit *Select buffer* auswählen, wenn Sie ihn schon zuvor eingelesen hatten.

Mit drei anderen Befehlen im *Window*-Menü können Sie die Fensteraufteilung des Bildschirms weiter beeinflussen. *Shrink window* und *Grow window* machen zum Beispiel das aktuelle Fenster um eine Zeile kleiner oder größer. Und mit dem Befehl *One window* können Sie erreichen, daß das aktuelle Fenster wieder den ganzen Bildschirm füllt und die anderen Fenster verschwinden (nicht jedoch der Inhalt der Puffer, der in ihnen angezeigt wurde).

9.3.9 Befehlsüberblick

Nach diesem »Mini-Schnell-Kurs« zu *MicroEmacs* sollten Sie mit diesem hervorragenden Editor mindestens alles das machen können, wozu Sie bislang *ed* verwendet haben. Wahrscheinlich sind aber jetzt noch mehr Fragen offen als geklärt wurden. Auf *MicroEmacs* noch intensiver einzugehen, wäre aber unfair gegenüber den anderen Themen, die in diesem Buch auch noch behandelt werden müssen. Vielleicht kann Ihnen aber die folgenden Befehlsübersicht helfen, zumindest einige der verbleibenden Fragen zu klären. Zu jedem Menübefehl von *MicroEmacs* finden Sie darin eine knappe Erläuterung, die zum Teil auf Aspekte des Programms eingeht, die bislang nicht besprochen wurden.

Das Project-Menü

Im *Project*-Menü finden Sie – wie auf dem Amiga allgemein üblich – Befehle, die ganze Dokumente oder das Programm selbst betreffen. Zum Beispiel die Befehle zum Öffnen neuer Textdateien oder zum Verlassen des Programms sind hier zu finden. Einen Befehl zum Drucken eines Textes werden Sie aber vergeblich suchen. Dazu können Sie die CLI-Befehle *type* oder *copy* (mit Ausgabeumlenkung auf *>PRT:*) verwenden.

Rename dient zum Umbenennen eines Puffers. Dies ist zunächst eine interne Aktion, die nichts weiter bewirkt, als daß dieser Puffer unter einem anderen Namen aufgelistet wird und zum Beispiel beim *Select buffer* unter dem neuen Namen angesprochen werden muß.

Open (Read) ersetzt den Inhalt des Puffers, der im aktuellen Fenster angezeigt wird, durch den Inhalt einer anderen Textdatei. Dazu müssen Sie den Namen dieser Datei (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile am unteren Bildschirmrand eingeben und mit [Return] abschließen. War der alte Pufferinhalt modifiziert und noch nicht gesichert worden, erhalten Sie eine Gelegenheit, den Befehl abzubrechen. (Antworten Sie dazu auf die Frage *Discard changes* mit [N] und [Return].)

Visit file arbeitet im wesentlichen exakt so wie *Open*. Die neu eingelesene Datei ersetzt allerdings nicht den Inhalt eines alten Puffers, sondern wird in einen neuen Puffer eingelesen.

Insert file fügt in den Inhalt einer anderen Textdatei eine Zeile vor der Zeile, in der sich die Schreibmarke befindet in den Puffer ein, der im aktuellen Fenster gezeigt wird. Dazu müssen Sie den Namen dieser Datei (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile am unteren Bildschirmrand eingeben und mit [Return] abschließen.

Save: Der Befehl *Save* sichert (schreibt) den Inhalt des Puffers, der im aktuellen Fenster gezeigt wird, auf Diskette beziehungsweise Festplatte.

Save as: Dieser Befehl sichert wie *Save* den Inhalt des Puffers, der im aktuellen Fenster gezeigt wird, auf Diskette beziehungsweise Festplatte. Sie können mit *Save as* einen Text jedoch unter einem anderen Namen sichern als dem, unter dem er eingelesen wurde. Dazu müssen Sie den neuen Namen (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile am unteren Bildschirmrand eingeben und mit [Return] abschließen.

Save mods arbeitet ebenfalls ähnlich wie *Save*, sichert jedoch alle Puffer, die in irgendeiner Form modifiziert wurden, nacheinander auf der Diskette beziehungsweise Festplatte.

Save/exit ist eine Kombination von *Save mods* und *Quit* (siehe unten).

New Cli erzeugt – ähnlich wie das CLI-Kommando *newcli* – ein neues CLI-Fenster. Es erscheint auf dem Workbench-Screen und Sie können damit ganz normal CLI-Kommandos verwenden, wie in jedem anderen CLI-Fenster auch. Wenn Sie dieses Fenster mit dem Kommando *endcli* schließen, kehren Sie zu *MicroEmacs* zurück.

Command: Nachdem Sie den Befehl *Command* eingegeben haben, können Sie in der Befehlszeile am unteren Bildschirmrand genau ein CLI-Kommando erteilen (zum Beispiel *dir df0:*). Die Ausgabe dieses Befehls wird dann auf dem Bildschirm aufgelistet und dahinter erscheint die Meldung »[End]«. Sie können sich nun diese Ausgabe in Ruhe betrachten. Wenn Sie mit dem *MicroEmacs* weiterarbeiten wollen, brauchen Sie nur [Return] betätigen und der Screen vom *MicroEmacs* mit dem bearbeiteten Text darin erscheint unverändert wieder am Bildschirm.

Quit beendet *MicroEmacs*. Falls modifizierte Puffer existieren, deren Inhalt noch nicht gesichert wurde, erhalten Sie aber die Gelegenheit, diesen Befehl abzubrechen.

Das Edit-Menü

Das *Edit*-Menü dient – wie der Name schon sagt – zum Editieren des Textes in einem Puffer. Sie finden in diesem Menü unter anderem eine Reihe von Befehlen zum Löschen, Kopieren und Duplizieren großer Textblöcke.

Kill region entspricht dem Befehl *Cut* des Werkzeugs *Notepad*. Er löscht den Bereich zwischen *dot* (Schreibmarke) und *mark* (einer Position, die Sie mit dem Befehl *Set Mark* festlegen) aus dem bearbeiteten Text (Puffer) und legt ihn in den Lösch-Puffer. Wenn Sie mehrfach hintereinander *Kill region* aufrufen, werden die so gelöschten Bereiche hintereinander in den Lösch-Puffer gelegt, bis Sie das erste Mal *Yank* (siehe unten) aufrufen.

Yank entspricht dem Befehl *Paste* des Werkzeugs *Notepad*. Er setzt den Inhalt des Lösch-Puffers vor der Schreibmarke in den gerade bearbeiten Text ein.

Set mark: Mit *Set Mark* oder einem Doppelklick mit der Maus wird das eine Ende des Bereichs markiert, der später zum Beispiel mit *Kill region* aus dem Text gelöscht werden kann.

Copy region entspricht dem Befehl *Copy* des Werkzeugs *Notepad*. Er legt den Bereich zwischen *dot* (Schreibmarke) und *mark* (einer Position, die Sie mit dem Befehl *Set Mark* festlegen) in den Lösch-Puffer. Der alte Inhalt des Lösch-Puffers wird dadurch ersetzt.

Upper region ändert die Groß-/Kleinschreibung des Bereichs zwischen Schreibmarke (*dot*) und Marke (*mark*). Alle Buchstaben darin sind nach Aufruf des Befehls groß geschrieben. (Das funktioniert allerdings nicht mit deutschen Umlauten!)

Lower region arbeitet wie *Upper region*; nur daß nach Aufruf alle Buchstaben des markierten Bereichs klein geschrieben sind.

List buffers öffnet ein neues Fenster, in dem alle schon geladenen Dateien (beziehungsweise ihre Puffer) aufgelistet werden.

Select buffer erlaubt die Auswahl eines anderen Puffers, der im aktuellen Fenster gezeigt werden soll. Sie müssen den Namen dieses anderen Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen.

Insert buffer fügt – ähnlich *Insert file* – den Inhalt eines anderen Puffers in den Puffer ein, der im aktuellen Fenster gezeigt wird. Sie müssen den Namen dieses anderen Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen.

Kill buffer entfernt einen Puffer aus dem Speicher. Sie müssen den Namen dieses Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen. Dieser Puffer darf dazu nicht (in einem oder mehreren Fenstern) sichtbar sein. Wurde er modifiziert und sind die Änderungen noch nicht gesichert, erhalten Sie die Gelegenheit, den Befehl abzubrechen. (Antworten Sie dazu auf die Frage *Discard changes* mit [N] und [Return].)

Justify buffer löscht in allen Zeilen des Puffers, der im aktuellen Fenster gezeigt wird, sämtliche Leerstellen und Tabulatoren, die sich am linken Rand befinden. Danach ist der Text in diesem Puffer also »linksbündig«.

Redisplay ist ein Befehl, der nur nach einem Programmfehler nötig sein sollte. Er zeichnet den gesamten Bildschirminhalt einmal neu (frischt ihn auf).

Quote Char unterdrückt die »spezielle Bedeutung« des folgenden Tastendrucks. Sie können damit Tasten und Tastenkombinationen in den Text einfügen, die normalerweise einen Befehl aufrufen würden (zum Beispiel ein [Ctrl]+[C]). Rufen Sie dazu erst *Quote Char* auf und drücken Sie dann die gewünschte Taste oder Tastenkombination. Solche »Control-Zeichen« oder »Escape-Sequenzen« innerhalb eines Textes werden manchmal zum Beispiel zur Druckersteuerung benötigt. Sie sind auch für die Eingabe von Makros wichtig (siehe unten).

Das Window-Menü

Das *Window*-Menü dient dazu, den Bildschirm, in dem *MicroEmacs* arbeitet, in eine Reihe verschiedener Bereiche (Fenster) zu unterteilen und diese Unterteilung zu ändern. In jedem dieser Fenster, die wenig mit den Intuition-Fenstern zu tun haben, wird der Inhalt eines Puffers angezeigt.

One window dehnt das aktuelle Fenster aus, bis es (fast) den ganzen Bildschirm füllt. Alle anderen Fenster verschwinden. Die Puffer, die in den anderen Fenstern angezeigt wurden, bleiben aber erhalten.

Split window teilt das aktuelle Fenster in zwei Hälften, in denen zunächst derselbe Puffer gezeigt wird.

Next window setzt die Schreibmarke in das nächste Fenster unterhalb des aktuellen Fensters.

Prev window setzt die Schreibmarke in das vorangehende Fenster oberhalb des aktuellen Fensters.

Grow window vergrößert das aktuelle Fenster auf Kosten eines benachbarten Fensters um eine Zeile.

Shrink window verkleinert das aktuelle Fenster um eine Zeile.

Next w-page blättert im nächsten Fenster unterhalb des aktuellen Fensters eine Seite weiter.

Prev w-page blättert im vorangehenden Fenster oberhalb des aktuellen Fensters eine Seite weiter.

Das Move-Menü

Das *Move*-Menü dient zur schnellen Verschiebung der Schreibmarke (schneller als mit den Cursortasten) zu gewissen »markanten Punkten« innerhalb eines Textes.

Top of buffer setzt die Schreibmarke in die erste Spalte der ersten Zeile des Puffers im aktuellen Fenster.

End of buffer setzt die Schreibmarke hinter den letzten Buchstaben der letzten Zeile des Puffers im aktuellen Fenster.

Top of window setzt die Schreibmarke in die erste Spalte der ersten Zeile des im aktuellen Fenster gezeigten Textes.

End of window setzt die Schreibmarke in die letzte Zeile des im aktuellen Fenster gezeigten Textes.

Go to line setzt die Schreibmarke in eine bestimmte Zeile. Dazu müssen Sie die Nummer dieser Zeile (wobei die erste Zeile im Puffer mit 1 numeriert wird) in der Befehlszeile eingeben und mit [Return] abschließen.

Swap dot&mark setzt die Schreibmarke an die Stelle der *Marke*, die zuvor mit *Set Mark* oder einem Doppelklick markiert wurde. Die Marke steht danach dort, wo zuvor die Schreibmarke gestanden hat.

Next page blättert das aktuelle Fenster eine Seite (Fensterhöhe) weiter.

Prev page blättert das aktuelle Fenster eine Seite (Fensterhöhe) zurück.

Next word setzt die Schreibmarke an den Anfang des Leerraums vor dem nächsten Wort im Text (auch wenn dieses sich erst in der folgenden Zeile befindet).

Previous word setzt die Schreibmarke an den Anfang des nächsten vorhergehenden Wortes. Befindet sich die Schreibmarke vor Aufruf dieses Befehls in einem Wort, steht sie danach auf dem Anfang dieses Wortes.

Scroll up verschiebt den im aktuellen Fenster angezeigten Text um eine Zeile nach oben. Die oberste Zeile verschwindet und am unteren Fensterrand wird Platz frei für eine der dort folgenden Zeilen.

Scroll down verschiebt den im aktuellen Fenster angezeigten Text um eine Zeile nach unten. Die unterste Zeile verschwindet und am oberen Fensterrand wird Platz frei für eine der vorangehenden Zeilen.

Das Line-Menü

Die Befehle im *Line*-Menü dienen unter anderem dazu, die Schreibmarke innerhalb einer Zeile zu bewegen, zeilenweise zu löschen, Zeilen aufzubrechen und die Schreibmarke auf und ab zu bewegen.

Open line spaltet die Zeile, in der sich die Schreibmarke befindet, in zwei Zeilen auf. Der Teil ab der Position der Schreibmarke kommt in eine neue Zeile. Insoweit kommt *Open line* einer Betätigung der [Return]-Taste gleich. Nach dem Aufteilen mit *Open line* steht die Schreibmarke aber am Ende der ersten Zeile, während sie nach [Return] am Anfang der neuen Zeile stehen würde.

Kill line löscht die komplette Zeile, in der sich die Schreibmarke gerade befindet, aus dem entsprechenden Puffer und legt sie in den Lösch-Puffer. (Erfolgen mehrere Löschoperationen hintereinander, ohne ein *Yank* oder *Copy region* dazwischen, wird der gelöschte Text hinter den alten Inhalt des Lösch-Puffers gehängt.)

Kill to eol löscht den Rest der Zeile, ab der Schreibmarkenposition, und legt ihn in den Lösch-Puffer (oder hängt ihn an den alten Inhalt des Löschpuffers an; siehe *Kill line*).

Start of line setzt die Schreibmarke an den Anfang der aktuellen Zeile.

End of line setzt die Schreibmarke an das Ende der aktuellen Zeile.

Prev line setzt die Schreibmarke eine Zeile höher.

Next line setzt die Schreibmarke eine Zeile tiefer.

Delete blanks löscht hinter der aktuellen Zeile alle Leerzeilen, so lange bis wieder eine Zeile auftaucht, die Buchstaben enthält.

Das Word-Menü

Das *Word*-Menü enthält Befehle, die Manipulationen an Worten (durch Leerstellen getrennte Buchstabenfolgen) entsprechen. Sie ähneln den entsprechenden Befehlen im *Line*-Menü. Beachten Sie aber bitte, daß sich auch im *Move*-Menü einige Befehle finden, die wortweise arbeiten.

Delete forw löscht den »Rest« (rechts von der Schreibmarke) des Wortes, in dem sich die Schreibmarke befindet. Als Ende eines Wortes gelten dabei Leerstellen, Tabulatoren, Satzzeichen wie Punkt und Komma, Control-Zeichen und das Zeilenende. Wenn sich die Schreibmarke in dem Leerraum vor einem Wort befindet, wird dieser Leerraum und das ganze folgende Wort gelöscht.

Delete backw arbeitet wie *Delete forw*. Nur wird nicht nach rechts zum Wortende, sondern in umgekehrter Richtung, zum Wortanfang hin, gelöscht.

Upper word wandelt den Rest des Wortes (rechts von der Schreibmarke), in dem sich die Schreibmarke befindet, in Großbuchstaben um. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

Lower word arbeitet wie *Upper word.* Nur werden die restlichen Buchstaben des Wortes in Kleinbuchstaben umgewandelt.

Cap. word wandelt den Buchstaben, auf dem sich die Schreibmarke befindet, in einen Großbuchstaben um und den Rest des Wortes (rechts von der Schreibmarke) in Kleinbuchstaben. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

Switch case wechselt die Groß-/Kleinschreibung aller Buchstaben, beginnend bei der Schreibmarke bis zum nächsten Wortende um. Aus allen Kleinbuchstaben werden Großbuchstaben und umgekehrt. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

Das Search-Menü

Das *Search*-Menü enthält Operationen zum schnellen Aufsuchen einer Textpassage in einem längeren Text und zum Austauschen einer häufig vorkommenden Textpassage gegen eine andere. Dieser Austausch kann entweder vollautomatisch oder erst nach vorheriger Bestätigung erfolgen.

Searchv forw sucht eine Textpassage, ausgehend von der aktuellen Position der Schreibmarke, nach vorne, also zum Ende des Textes hin. Dazu muß die gesuchte Textpassage nach Aufruf des Befehls in der Befehlszeile eingegeben und mit [Return] beendet werden. Ist die Suche erfolgreich, steht danach die Schreibmarke auf dem ersten Buchstaben hinter dem gefundenen Suchbegriff. Ist die Suche erfolglos, erscheint in der Befehlszeile die Meldung *Not found* (nicht gefunden).

Search back arbeitet genau wie *Search forw*; nur in umgekehrter Richtung, zum Anfang des Textes hin.

Search/rep. dient zum Austauschen einer Textpassage gegen eine andere im gesamten Text. Der Befehl verhält sich zunächst genau wie *Search forw*. Sobald der Suchbegriff das erste Mal gefunden worden ist, wird in der Meldungszeile (hinter dem Prompt *Replace:*) nach der Textpassage gefragt, die den Suchbegriff ersetzen soll. Geben Sie ihn ein und drücken Sie auf [Return]. Danach werden alle Vorkommnisse des Suchbegriffs, die hinter der aktuellen Position der Schreibmarke liegen, gegen den neu eingegebenen Text ersetzt.

Query s&r verhält sich wie *Search/rep.* Jedesmal, bevor der Suchbegriff gefunden wird, erscheint in der Befehlszeile jedoch die Frage *Change string [y/n/c]?*. Antworten Sie auf diese Frage mit der Taste [Y], wird der Suchbegriff ersetzt. Antworten Sie hingegen mit [N], wird der Suchbegriff nicht ersetzt, die Suche aber fortgesetzt. Antworten Sie mit [C], werden alle weiteren Vorkommnisse des Suchbegriffs ohne Rückfragen ersetzt. (*Query s&r* verhält sich dann wie *Search/rep.)* Wenn Sie die Suche abbrechen wollen, können Sie das jederzeit tun, indem Sie die Tastenkombination [Ctrl]+[G] drücken.

Das Extras-Menü

Das Extras-Menü enthält einige Befehle, die die Programmentwickler scheinbar in keinem anderen Menü sinnvoll unterbringen konnten. Es ist deshalb eine recht bunte Mischung von Befehlen.

Set Arg erlaubt es, eine Zahl einzugeben, die von einem unmittelbar darauffolgenden Befehl als »Argument« verwendet werden kann. Wenn Sie zum Beispiel den rechten Rand für den Text im aktuellen Fenster festlegen wollen, müssen Sie *Set Arg* aufrufen, die gewünschte Zahl in der Befehlszeile (hinter dem Prompt *Arg:*) eingeben und dann *Right mg* aufrufen. Drücken Sie nicht [Return], wenn Sie die Zahl eingegeben haben, und versuchen Sie auch nicht eine eventuell schon dort stehende Zahl mit [Backspace] zu löschen. Geben Sie einfach die gewünschte Zahl ein, und wählen Sie dann den Befehl, dem sie als Argument dienen soll!

Andere Befehle, die nicht unbedingt ein Argument benötigen, können Sie mittels *Set Arg* automatisch mehrfach durchführen lassen. Wählen Sie dazu *Set Arg*, geben Sie die gewünschte Zahl von Wiederholungen ein, und erteilen Sie dann den Befehl, der wiederholt werden soll. Sie können auf diese Weise zum Beispiel mit nur einem Befehl 100mal den Inhalt des Lösch-Puffers in den Text einsetzen lassen (mit *Yank* nach dem *Set Arg*). Sie können aber auch sehr schnell 20 Leerzeilen erzeugen, indem Sie *Set Arg* auswählen, »20« eintippen und dann auf die [Return]-Taste drücken.

Left mg dient dazu, einen linken Rand für einen Text festzulegen. Die Breite dieses Randes müssen Sie zuvor mit *Set Arg* angeben. Wenn Sie dann *Left mg* aufrufen, werden am linken Rand der aktuellen Zeile Leerstellen und Tabulatorzeichen eingefügt, um den durch *Set Arg* angegebenen Rand zu erreichen. Wenn Sie neue Zeilen eingeben, wird dieser Rand automatisch eingehalten, das heißt, am Anfang jeder neuen Zeile werden die entsprechenden Leerstellen und Tabulatoren eingefügt. Die Breite des linken Randes darf zwischen 0 und 80 Leerstellen liegen (60, wenn Sie die Workbench im 60-Zeichen-Modus betreiben).

Right mg dient dazu, die Breite (in Buchstaben) für den Text festzulegen. Dieser kann zwischen 1 und 80 liegen. Sie brauchen diese Zahl nicht vorher mit *Set Arg* anzugeben, sondern können sie auch nachträglich (nach Aufruf von *Right mg*) in der Befehlszeile eingeben. *MicroEmacs* stellt Zeichen, die rechts vom rechten Rand liegen, nicht mehr am Bildschirm dar. Ist eine Zeile länger, so wird als ihr letzter Buchstabe ein Dollarzeichen ($) gezeigt. Sie können daran sehen, daß diese Zeile länger ist als der von Ihnen bestimmte rechte Rand es erlaubt.

Word wrp erlaubt es eine Breite festzulegen, die eine Zeile bei der Eingabe maximal erreichen darf. Überschreitet ein Wort diese Grenze, rutscht es automatisch in eine neue Zeile. Diese Breite können Sie entweder vor Aufruf von *Word wrp* mit *Set Arg* festlegen oder nachträglich in der Befehlszeile eingeben. Beachten Sie bitte, daß es nicht sinnvoll ist, zu kleine Werte für *Word wrp* zu verwenden oder Werte, die größer sind als die mit *Right mg* festgelegte Zeilenbreite.

Save lvl dient dazu, *MicroEmacs* mitzuteilen, was getan werden soll, wenn das Programm beim Sichern eines Puffers feststellt, daß bereits eine Datei gleichen Namens existiert. Für das Verhalten von *MicroEmacs* in solchen Situationen gibt es drei Alternativen. Sie entsprechen

drei möglichen Werten des Arguments von *Save lvl*, das zuvor mit *Set Arg* angegeben werden muß (siehe oben). Normalerweise wird diese schon vorhandene Datei beim Sichern einfach überschrieben. Ist das Argument von *Save lvl* aber gleich -1, wird die schon vorhandene Datei unter dem Namen *edit.backup* in das Directory *t* kopiert und erst dann ersetzt. Ist aber das Argument von *Save lvl* gleich 0, so kann man mit *Save as* oder mit *Rename* keine schon vorhandene Datei überschreiben. Statt dessen erfolgt nur eine Fehlermeldung.

Start beginnt die Aufzeichnung eines sogenannten »Makros«. Jede Taste, die Sie betätigen nachdem Sie *Start* aufrufen, wird von *MicroEmacs* aufgezeichnet, bis Sie *End* aufrufen (siehe unten). Diese Tastenfolge ist ein Makro und kann später mit einem einzigen Befehl (*Execute*; siehe unten) wiederholt werden.

End beendet die Aufzeichnung eines Makros, die mit *Start* begonnen wurde (siehe oben).

Execute »führt das zuletzt aufgezeichnete Makro aus«. Das bedeutet, daß sich *MicroEmacs* dann verhält, als würden Sie noch einmal die Tasten drücken, die Sie zwischen *Start* und End *gedrückt* haben. Das gilt natürlich auch für Tastenkombinationen, die Befehlen entsprechen!

Set key dient dazu, eine beliebige Taste auf der Tastatur umzudefinieren (ähnlich, wie es das Werkzeug *SetMap* tut). Wählen Sie dazu *Set Key* aus dem Menü *Extras* und drücken Sie dann eine beliebige Taste, deren Bedeutung Sie ändern wollen. Sie können dann in der Befehlszeile eine beliebige Folge von Tasten eingeben (auch solche Tastenkombinationen, die Befehlen entsprechen). Schließen Sie die Tastenfolge mit [Return] ab. Wenn Sie später auf diese umdefinierte Taste drücken, werden die zuvor aufgezeichneten Tastendrücke »abgespielt«, als würden Sie nacheinander auf diese Tasten drücken. Die Umdefinition »gewöhnlicher« Tasten, wie [A], [,], [#] und so weiter, die verwirrend sein könnte, ist übrigens unmöglich. Umdefiniert werden können nur die Funktionstasten [F1] bis [F10] am oberen Rand der Tastatur und die Tasten des numerischen Blocks am rechten Rand.

Die Funktionstasten sind übrigens gleich nach dem Start schon mit solchen Tastenkombinationen belegt. Diese Belegung kann aber bei Ihnen unter Umständen von der in der folgenden Tabelle gezeigten etwas abweichen, da sich Commodore Änderungen am Programm vorbehält.

Befehl	**Wirkung**
[F1]	Aktuelle Zeile duplizieren.
[F2]	Aktuelle Zeile löschen.
[F3]	Makro ausführen.
[F4]	Eine Seite nach unten blättern.
[F5]	Eine Seite nach oben blättern.
[F6]	Aktuelles Fenster teilen (*Split window*).
[F7]	Aktuelles Fenster bildschirmfüllend machen (*One window*).
[F8]	Aktuelles Fenster eine Zeile nach oben schieben.
[F9]	Aktuelles Fenster eine Zeile nach unten schieben.
[F10]	Alle modifizierten Puffer sichern und Programm beenden.

Tabelle 9.1: *Belegung der Funktionstasten im* MicroEmacs

Load keys erlaubt es, eine ganze Reihe von Tasten-Umdefinierungen (wie mit Set key) auf einen Schlag vorzunehmen. Dazu müssen die Belegungen der entsprechenden Tasten in der richtigen Reihenfolge in eine Textdatei eingespeichert werden und mit dem Tilde-Zeichen (~) und einem [Return] dahinter voneinander getrennt werden. Die Reihenfolge, in der die Belegungen in dieser Textdatei stehen müssen, liegt fest. Zunächst kommt eine Zeile mit dem »auto-start-string«, anschließend 10 Zeilen mit den Belegungen der Funktionstasten [F1] bis [F10], dann die Belegung der [Help]-Taste und die Belegungen der Tasten im numerischen Tastenblock in der Reihenfolge [0], [1], [2], [3], [4], [5], [6], [.], [7], [8], [9], [Enter], [-].

Einer solchen Belegungsdatei kommt besondere Bedeutung zu, sofern sie »emacs_config« heißt und im Directory *SYS:config* liegt. Dieses Directory müssen Sie mit *makedir* erstellen. Es existiert normalerweise nicht auf der Workbench-Diskette. Findet *MicroEmacs* eine solche Datei, wird sie beim Start automatisch eingelesen (und die Funktionstasten dadurch umdefiniert). Am Ende dieses Einlesevorgangs wird dann der oben beschriebene »auto-start-string« ausgeführt, als würden Sie ihn über die Tastatur eingeben.

Interlace on schaltet den Screen von *MicroEmacs* in den Interlace-Modus um. Ihnen stehen dann 46 (stark flimmernde!) statt 21 Zeilen zur Verfügung, in denen Sie Texte bearbeiten können. Dieser Befehl ändert seinen Namen, wenn Sie ihn aufrufen. Er heißt danach *Interlace off* und schaltet bei erneutem Aufruf den Interlace-Modus wieder ab.

About ... zeigt Ihnen einen kleinen Requester mit Informationen über den Ursprung (zum Beispiel den Autor) und den augenblicklichen Stand dieser Version von *MicroEmacs*. Unter anderem erfahren Sie auch, daß Commodore Copyright auf diese Version beansprucht. Sie können den Requester mit einem Klick auf eine der beiden OK-Tasten wieder schließen.

9.4 Wie Sie das Werkzeug Notepad als Editor »mißbrauchen«

Wenn Ihnen die beiden eben beschriebenen Editoren aber zu umständlich oder schwierig zu erlernen sind, können Sie auch das *Notepad* als CLI-Editor mißbrauchen. Sie können *Notepad* dazu – wie jeden anderen Editor auch – vom CLI aufrufen mit:

```
1>notepad
```

Wenn Sie vermeiden wollen, daß *Notepad* das *Font*-Menü aufbaut und dazu lange die Diskette nach Schriftarten durchsucht, geben Sie dahinter einfach noch den Parameter -q an.

```
1>notepad -q
```

Und wenn Sie gleich eine Datei laden wollen, um Sie zu betrachten oder zu ändern, können Sie deren (Pfad-) Namen auch gleich beim Start angeben.

```
1>notepad -q S:Startup-sequence
```

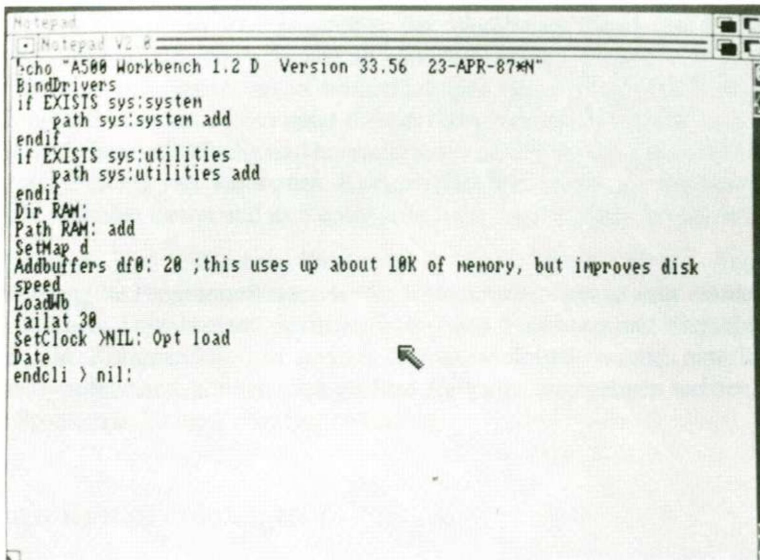


Bild 9.4: *Die CLI-Textdatei Startup-Sequence im Notepad*

Sie können Textdateien aber natürlich auch nachträglich einlesen, indem Sie den Befehl *Open* aufrufen. Achten Sie aber bitte darauf, daß Sie in dem daraufhin erscheinenden Requester den vollen Pfadnamen der zu ladenden Datei angeben müssen, wenn diese nicht im aktuellen Directory liegt.

Ansonsten können Sie mit dem *Notepad* Textdateien bearbeiten wie mit jedem anderen Editor auch. Verwenden Sie aber bitte keine verschiedenen Schriftarten und Schriftstile in einer

solchen Datei. Sie können Textdateien mit verschiedenen Schriftarten zum Beispiel nicht mehr als Kommandofolgen verwenden. Wenn Sie in einem längeren Text nicht mehr sicher sind, ob Sie denn nun Schrift- oder Stilwechsel vorgenommen haben oder nicht, können Sie sichergehen, indem Sie die beiden Befehle *Remove Fonts* und *Remove Styles* im *Format*-Menü auswählen, bevor Sie den Text wieder abspeichern.

10 Einfacher arbeiten mit Kommandofolgen

Ein großer Vorteil, den das CLI gegenüber der Workbench bietet, ist die Möglichkeit, langwierige, häufig anfallende Aufgaben zu »automatisieren«. Hierzu wird eine Folge von CLI-Befehlen zu einer sogenannten »Kommandofolge« zusammengefaßt. Eine Kommandofolge erhält einen Namen und kann unter diesem Namen durch den Befehl *execute* gestartet werden. Sie läuft dann selbständig und normalerweise ohne weiteres Zutun des Benutzers ab. Mit Hilfe des Befehls *run* kann man Kommandofolgen sogar als separaten Task »im Hintergrund« ablaufen lassen und sich unterdessen um andere Dinge kümmern.

Kommandofolgen sind einfachen Programmen sehr ähnlich. Keine Angst vor der Programmierung; die Programmiersprache für Kommandofolgen ist sehr einfach! Sie kennt aber schon bedingte Anweisungen, Sprunganweisungen, Parameter und Variablen. Durch die Möglichkeit, in Kommandofolgen andere Kommandofolgen – und natürlich normale Kommandos – aufzurufen, können auch größere Aufgaben angegangen werden, indem man ein großes Problem in kleine Teilaufgaben zerlegt.

10.1 Wozu Kommandofolgen ?

Der Sinn von Kommandofolgen wird Ihnen vielleicht nicht gleich klar sein. Man benötigt sie üblicherweise erst dann, wenn man intensiv mit dem Computer arbeitet. Sie werden rasch merken, daß viele Dinge, die Sie täglich tun, sich immer wiederholen oder sich zumindest ähneln.

Nehmen wir einmal an, Sie würden häufiger eine Datei erst sortieren, dann ausdrucken und dazu immer die drei folgenden CLI-Befehle hintereinander ausführen:

```
sort   kundenliste TO druckdatei
type   druckdatei  TO PRT:
delete druckdatei
```

Dann ist es doch eigentlich recht lästig, daß Sie jedesmal alle drei Befehle fehlerfrei eintippen müssen, nach jedem Befehl darauf warten müssen, daß er fertig wird und alles in allem eine stupide mechanische Tätigkeit ausführen müssen, für die ein Computer viel besser geeignet ist. Schöner wäre es, diesen Vorgang mit einem Befehl starten zu können und sich dann anderen Aufgaben zuwenden zu können.

Genau das ermöglichen Kommandofolgen! Wie Sie im folgenden sehen werden, können Sie eine Kommandofolge schreiben, die dasselbe leistet wie die drei oben gezeigten Zeilen und nur mit einem einzigen Befehl aufgerufen werden kann. Falls die Datei, die sortiert und gedruckt werden soll, nicht immer denselben Namen hat, kann die Kommandofolge auch daran flexibel angepaßt werden. Kurz: Kommandofolgen sparen Tipparbeit, machen Fehler unwahrscheinlicher und nehmen uns stupide Arbeit ab!

10.2 Eine einfache Kommandofolge

Das Format von Kommandofolgen ist denkbar einfach. Man tippt die CLI-Befehle in genau derselben Form, wie man sie auch im CLI-Fenster eingeben würde, in eine Textdatei. Idealerweise macht man dies mit einem Texteditor. Im letzten Kapitel wurden ja zwei Texteditoren (*ed* und *MicroEmacs*) etwas ausführlicher vorgestellt. Sie können aber natürlich auch *Notepad* verwenden, wenn Sie damit vertrauter sind.

Für die folgenden Versuche starten Sie nun bitte das CLI – falls Sie sich noch nicht darin befinden. Richten Sie mit *makedir df0:test* ein neues Dateiverzeichnis namens *test* ein. Danach machen Sie bitte *test* mit *cd df0:test* zum aktuellen Dateiverzeichnis. Dieses Dateiverzeichnis wird die Dateien aufnehmen, die für die folgenden Versuche benötigt werden. Der Befehl *execute*, mit dem Sie Kommandofolgen ja starten müssen, »findet« Kommandofolgen nämlich nur im aktuellen Dateiverzeichnis und im logischen Laufwerk *S:*. Kommandofolgen in anderen Directories müssen Sie mit dem vollen Pfadnamen hinter *execute* aufrufen. Wenn Sie später »ernsthafte« Kommandofolgen schreiben, sollten Sie diese deshalb immer in das logische Laufwerk *S:* legen. Für Ihre ersten Versuche in dieser Hinsicht ist ein separates Directory aber besser, da Sie es später leicht mit einem *delete*-Befehl wieder löschen können.

Sie werden nun Ihre erste Kommandofolge erstellen. Verwenden Sie dazu den Texteditor Ihrer Wahl. Falls Sie beim Eintippen der folgenden Zeilen Fehler machen sollten, benutzen Sie bitte die Korrekturmöglichkeiten des jeweiligen Editors. Starten Sie jetzt den Editor Ihrer Wahl mit **einem** der folgenden drei Befehle:

```
1>ed ZeigeText
```

```
1>microemacs ZeigeText
```

```
1>notepad
```

Tippen Sie dann die folgenden Zeilen ein:

```
echo "Start der Kommandofolge"
copy text TO kopie
```

```
type kopie OPT N
delete kopie
echo "Kommandofolge beendet!"
```

Sichern Sie dann Ihr Werk (wenn Sie Notepad verwenden, müssen Sie dazu den Namen »ZeigeText« angeben) und verlassen Sie den Editor. Wenn Sie zurück im CLI sind und *cd* eingeben, werden Sie Ihre erste Kommandofolge unter dem Namen *ZeigeText* angezeigt bekommen. Geben Sie nun den Befehl *execute ZeigeText* ein und die Kommandofolge *ZeigeText* wird gestartet. Sie wird aber schon in der zweiten Zeile abgebrochen, da der Befehl *copy text TO kopie* nicht funktionieren kann. Es gibt ja keine Datei namens *text* im aktuellen Dateiverzeichnis, die kopiert werden könnte. Diese müssen Sie erst einmal erzeugen.

Gehen Sie zurück in den Editor und geben Sie ein paar kurze Zeilen ein, die Sie unter dem Namen »text« abspeichern. Wenn Sie sich Arbeit ersparen wollen, können Sie auch eine vorhandene Textdatei (zum Beispiel S:Startup-Sequence) unter dem Namen »text« in das aktuelle Directory kopieren. Starten Sie dann erneut Ihre Kommandofolge mit *execute ZeigeText*. Es sollten nun die folgenden Zeilen auf Ihrem Bildschirm erscheinen:

```
1>execute ZeigeText
Start der Kommandofolge
"Hier erscheinen die von Ihnen einge-"
"gebenen Zeilen der Datei text mit"
"vorangestellten Zeilennummern ......"
Kommandofolge beendet!
1>
```

Die Kommandofolge *ZeigeText* hat also soeben selbständig und ohne Ihr Zutun fünf CLI-Befehle ausgeführt. Nicht schlecht für den Anfang! Nur ist die gezeigte Kommandofolge noch sehr unflexibel. Sie kann nur eine einzige Datei, die den Namen *text* hat, ausgeben. Wollen Sie einmal eine andere Datei ausgeben, so müßten Sie dafür eine neue Kommandodatei schreiben, die sich kaum von der ersten unterscheiden würde. Nur der Name *text* müßte bei jeder dieser neuen Kommandodateien durch einen anderen ersetzt werden. Eine sehr lästige Angelegenheit.

Gott sei Dank kann aber der *execute*-Befehl diese Ersetzung für Sie übernehmen. Sie werden nun eine Kommandofolge schreiben, die eine beliebige Datei ausgeben kann, deren Namen Sie der Kommandofolge beim Aufruf als *Parameter* übergeben müssen.

10.3 Kommandofolgen mit Parametern

Um Parameter in Ihre Kommandofolge »einzubauen«, müssen Sie zwei Dinge tun. Sie müssen am Anfang der Kommandofolge dem *execute*-Befehl mitteilen, welche Parameter mit welchen Namen die Kommandofolge benötigt. In der Kommandofolge selbst muß dann noch der Name *text* gegen einen »Platzhalter« ersetzt werden. Diesen Platzhalter ersetzt *execute* dann, bevor

die Kommandofolge wirklich ausgeführt wird, gegen den Namen, den Sie beim Aufruf der Kommandofolge angeben. Die Kommandofolge *ZeigeText* sieht dann aus wie folgt:

```
.KEY datei/a
echo "Start der Kommandofolge"
copy <datei> TO kopie
type <datei> OPT N
delete kopie
echo "Kommandofolge beendet!"
```

Geben Sie diesen Text nun mit dem Editor Ihrer Wahl ein oder ändern Sie die vorhandene Kommandofolge entsprechend ab. Auch diese geänderte Datei speichern Sie dann bitte unter dem Namen *ZeigeText*.

Wie Sie sehen, taucht in dieser Kommandofolge überall, wo vorher *text* stand, nun der Platzhalter *<datei>* auf. Ferner ist eine zusätzliche erste Zeile eingefügt worden, die angibt, welche Parameter die Kommandofolge benötigt. Führen Sie nun bitte die nötigen Änderungen an der Datei *ZeigeText* mit Hilfe von *ed* durch.

Die neue erste Zeile von *ZeigeText* sieht deutlich anders aus als die restlichen Zeilen. Sie beginnt zum Beispiel mit einem Punkt, was sie als eine »besondere« Zeile kennzeichnet, die nicht als Befehl ausgeführt werden darf, sondern spezielle Informationen für *execute* enthält. *Execute* sucht vor der Ausführung jeder Kommandofolge nach solchen Zeilen und wertet sie aus. Das Wort *KEY* nach dem Punkt bedeutet, daß der Rest der Zeile die Beschreibung eines Parameters enthält. In diesem Fall lautet sie *.KEY datei/a*, was bedeutet, daß *ZeigeText* einen Parameter namens *datei* hat und dieser Parameter notwendig ist, also beim Aufruf nicht fortfallen darf. Letzteres regelt der Zusatz */a*, der nicht zum eigentlichen Namen des Parameters gehört. Stände hinter *datei* kein */a*, so wäre *datei* ein optionaler Parameter, was unter Umständen auch recht sinnvoll sein kann. Mehr darüber aber weiter unten.

Probieren Sie nun einmal das neue *ZeigeText* aus. Wenn Sie einfach nur *execute zeigetext* eingeben, wie bisher, erhalten Sie sofort eine Fehlermeldung, die besagt, daß Sie einen Parameter vergessen haben. Tippen Sie aber *execute ZeigeText text*, so erhalten Sie am Bildschirm dasselbe Ergebnis wie früher mit *execute ZeigeText*. Probieren Sie auch einmal *execute ZeigeText ZeigeText* aus. Ihnen wird dann am Bildschirm der Inhalt der Kommandofolge selbst gezeigt.

Kommen wir nun aber zum trockeneren Teil dieses Kapitels. Bisher haben Sie ja nur am Beispiel gesehen, wie man Parameter verwendet. Nun sollen Ihnen auch alle Details offenbart werden.

Wie schon oben zu sehen, gehört zu jeder Kommandofolge mit Parametern eine KEY-Zeile. Eine solche Zeile teilt *execute* mit, welche Parameter benötigt werden, wie sie heißen, wie sie behandelt werden müssen und ob sie optional oder notwendig sind. In einer KEY-Zeile können gleichzeitig mehrere Parameter spezifiziert werden. Die einzelnen Parameter werden dann durch Kommata getrennt. Die folgende Zeile spezifiziert zum Beispiel drei (optionale) Parameter:

```
.KEY datei,disk,name
```

Hinter dem Namen jedes Parameters können, von einem Schrägstrich getrennt, zwei verschiedene Buchstaben stehen, die angeben, wie der Parameter zu behandeln ist. Der Zusatz /*A* bedeutet, daß der Parameter nötig ist, also beim Aufruf unbedingt angegeben werden muß (mehr zu optionalen Parametern weiter unten). Wird er beim Aufruf vergessen, ergibt das eine Fehlermeldung und die Kommandofolge wird abgebrochen.

Der Zusatz /*K* bedeutet, daß der Parameter immer von seinem Namen eingeleitet werden muß. Würde die erste Zeile von ZeigeText .KEY datei/A/K lauten, so wäre *execute ZeigeText text* kein legaler Aufruf der Kommandofolge mehr. Die korrekte Schreibweise wäre nun *execute ZeigeText DATEI text* (die Großschreibung von *datei* dient nur der besseren Lesbarkeit).

Der Sinn dieser auf den ersten Blick etwas umständlichen Einrichtung liegt in der besseren Lesbarkeit eines Aufrufs – besonders, wenn optionale Parameter ins Spiel kommen. Hat eine Kommandofolge drei optionale Parameter und Sie übergeben nur zwei, hat es *execute* nämlich schwer, »sich zu entscheiden«. Haben Sie den ersten und zweiten Parameter übergeben und den dritten freigelassen? Oder handelt es sich um den ersten und dritten und der zweite soll freibleiben? Oder sind es gar der zweite und dritte Parameter und der erste bleibt unbelegt? Schreiben Sie aber vor die beiden Parameter auch noch ihre Namen, wird die Sache eindeutig. Sie können dann sogar die Reihenfolge austauschen und erst den zweiten, dann den ersten Parameter angeben, usw. Die Namen machen die ganze Sache eindeutig.

Nachdem Sie einen oder mehrere Parameter spezifiziert haben, können diese Parameter in der Kommandofolge innerhalb eines CLI-Kommandos verwendet werden. An den Stellen, an denen der »Wert« des Parameters erscheinen soll, müssen Sie den Namen des Parameters in spitzen Klammern eingeben; zum Beispiel *<datei>* für den Parameter *datei*. Der Wert eines Parameters ist der Text, der beim Aufruf anstelle des Parameters eingegeben wurde. Vor der Ausführung der Kommandofolge wird dann jedes Vorkommen des Platzhalters *<datei>* gegen den Wert *text* ersetzt, wenn Sie *execute ZeigeText text* eingegeben haben.

Problematisch wird diese Ersetzung bei optionalen Parametern, für die kein Wert angegeben wurde. Gegen welchen Text sollen die entsprechenden Platzhalter ersetzt werden? Wenn Sie nichts »Besonderes« unternehmen, ersetzt *execute* solche Platzhalter gegen den leeren Text "". Dies dürfte manchmal sinnvoll sein, oft aber auch nicht. Hierfür gibt es eine besondere Einrichtung, die sogenannten *Defaultwerte*. Ein Defaultwert wird immer dann für einen Platzhalter eingesetzt, wenn dem entsprechenden Parameter beim Aufruf der Kommandofolge kein Wert zugeteilt wurde. Es gibt zwei Möglichkeiten, einem Parameter einen Defaultwert zuzuteilen.

Die Zeile *.DEF datei text* in einer Kommandofolge ordnet dem Parameter *datei* z.B. den **Def**aultwert *text* zu. Das allgemeine Format einer solchen *DEF*-Zeile sieht dabei aus wie folgt:

```
.DEF <Parameter> <Defaultwert>
```

Jedes Vorkommen des entsprechenden Platzhalters <Parameter> wird dann, natürlich nur, falls kein anderer Wert beim Aufruf übergeben wurde, gegen den Defaultwert ersetzt. Flexibler

ist die Möglichkeit, direkt beim Platzhalter einen Defaultwert anzugeben. Der Platzhalter eines Parameters kann so an unterschiedlichen Stellen einen unterschiedlichen Defaultwert haben. Hierzu schreibt man hinter dem Namen des Parameters im Platzhalter den Defaultwert, getrennt durch ein Dollarzeichen. *<datei$text>* hat also für diesen einen Platzhalter dieselbe Wirkung wie *.DEF datei text* für die ganze Kommandofolge. *<datei>* wird durch *text* ersetzt, falls beim Aufruf kein anderer Wert für den Parameter *datei* übergeben wurde.

Diese Möglichkeit soll jetzt aber sofort für die erste Beispiel-Kommandofolge eingesetzt werden. Hierzu wird sie nun geändert, bis sie das folgende Aussehen hat:

```
.KEY datei
.DEF datei text
echo "Start der Kommandofolge"
copy <datei> TO kopie
type <datei> OPT N
delete kopie
echo "Kommandofolge beendet!"
```

Es muß also in der ersten Zeile der alten Datei das /a gelöscht werden und die zweite Zeile so eingefügt werden, wie sie hier gezeigt wurde. Falls Sie nicht mehr wissen, wie die Änderungen auszuführen sind, lesen Sie die dafür nötigen Befehle und Handgriffe einfach in der Beschreibung des jeweiligen Texteditors, also im letzten Kapitel beziehungsweise in der Beschreibung von *Notepad*, nach.

Nun sollten Sie Ihre neue und verbesserte Kommandofolge aber auch ausprobieren. Sichern Sie dazu die geänderte Fassung von *ZeigeText*, verlassen Sie den Texteditor und geben Sie dann *execute ZeigeText* im CLI ein. Wie in der allerersten Version der Kommandofolge *ZeigeText* wird Ihnen nun der Inhalt der Datei *text* gezeigt. Tippen Sie aber *execute ZeigeText S:startup-sequence*, so wird der Inhalt von *startup-sequence* angezeigt. Defaultwerte sind ganz schön praktisch, nicht wahr?

10.4 Spezielle Punktbefehle

Damit wissen Sie nun schon fast alles, was über Parameter zu sagen wäre. Außer den zwei Punktbefehlen *.KEY* und *.DEF* gibt es aber noch eine Reihe anderer Punktbefehle, die zwar nicht häufig benötigt werden, der Vollständigkeit halber nun aber auch noch aufgelistet werden. Wie *.KEY* und *.DEF* müssen auch sie alle am Beginn einer neuen Zeile stehen (der Punkt muß also der erste Buchstabe der Zeile sein), sonst erkennt *execute* sie nicht als Punktbefehle.

.K ist eine Abkürzung für *.KEY*.

.DOT und dahinter ein einzelner Buchstabe, führt für den Rest der Kommandofolge dazu, daß nicht mehr der Punkt, sondern der hier angegebene Buchstabe zur Einleitung von Punktbefehlen dient. Nach *.DOT!* wird also *.DEF* nicht mehr erkannt; man muß statt dessen *.DEF* schreiben.

.BRA, und dahinter ein einzelner Buchstabe, führt für den Rest der Kommandofolge dazu, daß nicht mehr das Kleiner-Zeichen < zur Einleitung von Platzhaltern dient. Statt dessen müssen Platzhalter für Variablen jetzt mit dem angegebenen Buchstaben beginnen. Nach *.BRA {* wird also *<datei>* nicht mehr als Platzhalter erkannt; man muß statt dessen *{datei>* schreiben.

.KET, und dahinter ein einzelner Buchstabe, führt für den Rest der Kommandofolge dazu, daß nicht mehr das Größer-Zeichen> zur Beendigung von Platzhaltern dient. Statt dessen müssen Platzhalter für Variablen jetzt mit dem angegebenen Buchstaben enden. Nach *.KET }* wird also *<datei>* nicht mehr als Platzhalter erkannt; man muß statt dessen *<datei}* schreiben. Stehen am Anfang einer Kommandofolge also (in zwei getrennten Zeilen!) die beiden Befehle *.BRA {* und *.KET },* müssen in den darauffolgenden Zeilen Platzhalter in geschweifte und nicht in spitze Klammern eingeschlossen werden.

.DOL oder *.DOLLAR,* und dahinter ein einzelner Buchstabe, führen für den Rest der Kommandofolge dazu, daß nicht mehr das Dollar-Zeichen *$* zur Trennung von Parameternamen und Defaultwerten in Platzhaltern dient. Statt dessen müssen Parameternamen und Defaultwerte in Platzhaltern jetzt durch den angegebenen Buchstaben getrennt werden. Nach dem Befehl *.DOL * wird also *<datei$text>* nicht mehr als Platzhalter mit Defaultwert erkannt; man muß statt dessen *<datei\text>* schreiben.

.<Leerzeichen>, also ein einzelner Punkt am Anfang der Zeile, gefolgt von einem Leerzeichen, erklären den Rest der Zeile zum Kommentar, den weder *execute* noch das CLI berücksichtigen. Endet die Zeile sofort hinter dem Punkt, bedeutet dies ebenfalls einen Kommentar – in diesem Fall ist dieser Kommentar eine Leerzeile, die der besseren Lesbarkeit halber eingefügt wurde.

Schauen Sie sich diese Punktbefehle sorgfältig an! Bis auf die Kommentare werde ich sie zwar in den folgenden Kommandofolgen nicht verwenden, sollten Sie aber einmal eine Kommandofolge lesen müssen, die jemand anderes geschrieben hat, ist es gut, diese Befehle zu kennen.

10.5 Bedingte Anweisungen und Sprünge

Im vorangegangenen Abschnitt haben Sie schon eine recht komfortable Kommandofolge konstruiert. Sie hat aber immer noch einige Mängel oder Schwächen. Wäre es zum Beispiel nicht besser, wenn die Kommandofolge auch selbst merken würde, ob die Datei, die sie kopieren und ausdrucken will, auch existiert und eine entsprechende Fehlermeldung ausgeben würde, falls nicht? Im Augenblick bricht die Kommandofolge *ZeigeText* in diesem Fall nähmlich nur mit einer (englischen!) Fehlermeldung ab. In diesem Abschnitt werden Sie erfahren, wie Sie Fehler, die bei der Ausführung einer Kommandofolge auftreten, »abfangen« und dem Benutzer sinnvoll melden können.

Die erste Kommandofolge, die Sie bis jetzt kennengelernt haben, ist zudem extrem geradlinig. Egal was geschieht, folgt sie stur der Abfolge von CLI-Befehlen, wie Sie sie eingegeben haben. Dies ist nicht immer wünschenswert. Wenn während der Ausführung einer Kommandofolge zum Beispiel ein Fehler auftritt, kann es sinnvoll sein, die Kommandofolge nicht einfach abzubrechen. Manche Fehler können vielleicht ausgebügelt werden. Zumindest sollte dem

Benutzer in sinnvoller Form die Fehlerursache mitgeteilt werden. Unter Umständen kann es auch sinnvoll sein, zwei Fliegen mit einer Klappe zu schlagen, wenn man eine Kommandofolge schreibt – also zwei verschiedene Aufgaben zu erledigen, je nachdem welche Parameter man ihr übergibt. Beides ist nur dann möglich, wenn man Entscheidungspunkte in eine Kommandofolge einbauen kann, an denen man, abhängig von gewisssen Bedingungen, entscheiden kann, ob man die eine oder andere Folge von Befehlen durchführt.

Zunächst soll die Kommandofolge *ZeigeText* deshalb nachsehen, ob die auszugebende Datei überhaupt existiert. Die dafür nötige Änderung ist relativ simpel. Sie zeigt eine einfache Anwendung des *if*-Kommandos (engl. *if = wenn/falls*):

```
.KEY datei
.DEF datei text
echo "Start von ZeigeText"
IF EXISTS <datei>
copy <datei> TO kopie
type <datei> OPT N
delete kopie
echo "ZeigeText erfolgreich beendet!"
ELSE
echo "Fehler: <datei> existiert nicht!"
ENDIF
```

(Die für Kommandofolgen spezifischen Befehle wie *if* werden übrigens in allen Auflistungen von Kommandofolgen von nun an immer groß geschrieben, um ihre Bedeutung hervorzuheben. Wenn Ihnen das besser gefällt, können Sie diese Befehle natürlich klein schreiben – dem CLI ist's egal.) *ZeigeText* zeigt nun immer noch den Inhalt der Datei im CLI-Fenster, deren Name als Parameter übergeben wird – falls es diese Datei überhaupt gibt. Falls Sie nicht existiert, bricht nicht etwa die Kommandofolge ab, sondern es wird eine (hoffentlich) verständliche Fehlermeldung ausgegeben, die dem Anwender das Problem verdeutlicht. Der normale Ablauf der Kommandofolge, wie wir ihn bisher kannten, wird dazu in eine *if*-Anweisung eingeschlossen. Obwohl *if*, *else* und *endif* einzelne CLI-Kommandos sind, werden sie immer zusammen verwendet und sollen deshalb auch hier zusammen als »*if*-Anweisung« erläutert werden. Das Format einer *if*-Anweisung lautet:

```
IF <Bedingung>
 Folge von CLI-Befehlen...
 ...
ELSE
 andere Folge von CLI-Befehlen...
 ...
ENDIF
```

Wenn die Liste der CLI-Befehle hinter *else* leer ist, so kann man *else* auch ganz wegfallen lassen und die Anweisungsfolge sofort mit *endif* beenden. Die Vorgehensweise von *execute*, wenn bei der Ausführung einer Kommandofolge auf ein *if* getroffen wird, ist nun die folgende:

Die Bedingung hinter dem *if* wird überprüft. Falls sie zutrifft, werden die Anweisungen hinter dem *if* ausgeführt, bis ein *else* oder *endif* gefunden wird, je nachdem, was zuerst eintritt. Trifft *execute* auf ein *else*, so werden alle Kommandos dahinter bis zum *endif* ignoriert. Hinter dem *endif* wird aber auf alle Fälle wieder mit der Ausführung der Kommandos fortgefahren.

Trifft die Bedingung hinter *if* nicht zu, so werden alle Anweisungen ignoriert, bis man auf ein *else* oder *endif* trifft. Dahinter wird auf alle Fälle wieder mit der Ausführung der Kommandos fortgefahren. Eine *if*-Anweisung hat also zwei »Zweige«, die **alternativ** durchlaufen werden. Entweder die Kommandos in dem Zweig hinter dem *if* oder die Kommandos in dem anderen Zweig hinter dem *else* werden durchgeführt. Fällt der Zweig hinter dem *else* ganz fort, so kann man auch von einer *bedingten Anweisungsfolge* sprechen. Nur wenn die Bedingung zutrifft, wird sie ausgeführt – sonst nicht.

In unserem Beispiel sieht die geprüfte Bedingung umgangssprachlich etwa folgendermaßen aus: Wenn (*if*) die auszugebende Datei vorhanden ist, kopiere sie und gib sie aus, sonst (*else*) melde einen Fehler (*endif*). Die Bedingung *EXISTS <datei>* ist dabei nur eine von mehreren Bedingungen, die Sie hinter *if* verwenden können.

Weitere Bedingungen sind zum Beispiel *WARN*, *ERROR* und *FAIL*, die dann zutreffen, wenn der letzte CLI-Befehl fehlschlug und der letzte *Fehlercode* größer oder gleich 5 (bei *WARN*), 10 (bei *ERROR*) oder 20 (bei *FAIL*) ist. Jeder fehlgeschlagene CLI-Befehl setzt einen solchen Fehlercode (für den man mittels *why* ja auch eine Erläuterung erhalten kann). Mit den Bedingungen *WARN*, *ERROR* und *FAIL* kann man abprüfen, ob ein Fehler passiert ist, und dann geeignete Gegenmaßnahmen ergreifen oder wenigstens den Benutzer informieren. Bei allen drei Bedingungen muß man aber berücksichtigen, daß sie abhängig von dem aktuellen Abbruch-Level sind, der mit *failat* geändert werden kann. Ist der Fehlercode größer als der aktuelle Abbruch-Level, so wird die Ausführung einer Kommandofolge sofort abgebrochen und eine eventuell folgende *if*-Anweisung wird überhaupt nicht mehr ausgeführt.

Steht der Abbruch-Level beispielsweise bei 5 und es tritt ein Fehler ein, der einen Fehlercode von 10 hat, so bricht jede Kommandofolge ab. Vor jedem Befehl, der einen Fehler verursachen könnte, den man selbst behandeln will, sollte man deshalb mit *failat* den Abbruch-Level hochsetzen. Mehr dazu aber weiter unten.

Eine weitere Bedingung erlaubt den Vergleich zweier Texte und trifft dann zu, wenn beide gleich sind. Beim Vergleich wird Groß- oder Kleinschreibung ignoriert; »A« ist gleich »a«. Hiermit wird man üblicherweise wohl die Werte von Parametern mit bestimmten Texten vergleichen und zwei Parameter miteinander. Das Format für einen solchen Vergleich lautet: *text1 EQ text2* (*EQ* steht dabei für *equal* oder auf deutsch *gleich*). Ein möglicher Vergleich dieser Art wäre zum Beispiel *IF<datei> EQ "text"*. Diese Bedingung würde genau dann zutreffen, wenn der Wert des Parameters *datei* gleich »Text« ist.

Und schließlich kann man auch noch die Bedeutung einer Bedingung umkehren, indem man direkt hinter IF das Schlüsselwort NOT (engl. für *nicht*) setzt. *IF NOT "text" EQ "text"* trifft zum Beispiel nie zu und führt dazu, daß der erste Zweig der entsprechenden *if*-Anweisung nicht ausgeführt wird.

Bislang war die Erklärung der *if*-Anweisung allerdings etwas vereinfacht. *If*-Anweisungen können nämlich geschachtelt sein. Deshalb gehört nicht immer das unmittelbar nächste *else* zu einem *if*, wenn im *if*-Zweig erneut ein *if*-Befehl auftaucht.

```
IF <Bedingung>              ; äußeres IF
 andere CLI-Befehle...
.
  IF <Bedingung>            ; inneres IF
    andere CLI-Befehle...
  ELSE                      ; inneres ELSE
    andere CLI-Befehle...
  ENDIF                     ; inneres ENDIF
.
ELSE                        ; äußeres IF
 andere CLI-Befehle...
 ...
ENDIF                       ; äußeres IF
```

Die Zuordnung von einem *else* oder *endif* zu einem bestimmten *if* ist ähnlich wie mit geschachtelten Klammern und wohl auch recht intuitiv zu verstehen.

Eine *if*-Anweisung kann unter Umständen auch als ein Sprung beziehungsweise ein Überspringen von Teilen einer Kommandofolge aufgefaßt werden. Trifft eine Bedingung nicht zu, werden ja die Anweisungen zwischen *if* und *else* oder *endif* übersprungen. Eine etwas direktere Möglichkeit, Anweisungen zu überspringen, stellt das *skip*-Kommando dar. Trifft man innerhalb einer Kommandofolge auf ein *skip*, so bedeutet dies ein Überspringen (engl. *skip*) aller Kommandos in der Folge, bis zu einem *lab*-Kommando (*lab* für *label*; zu deutsch *Marke*). Sowohl *skip* als auch *lab* haben einen Parameter, der aus einem beliebigen Text bestehen kann. *Skip* springt immer bis zu einem *lab*, das denselben Parameter hat. Dazwischenliegende *labs* mit anderen Parametern werden ebenfalls übersprungen, wie alle anderen Befehle auch.

```
SKIP Ende
 bel. CLI-Befehle...
 ...
LAB  Ende                   ; Sprung bis hier
 bel. CLI-Befehle...
```

Die letzte »Sprunganweisung«, die Sie in Kommandofolgen verwenden können, ist schließlich *quit*. Dieser Befehl stellt einen recht »brutalen« Sprung dar, mit dem die gesamte Kommandofolge verlassen wird, als wäre ihr Ende erreicht worden. Der Befehl *quit* besitzt einen optionalen Parameter, der ein Fehlercode (eine Zahl) sein sollte. Ist er nicht vorhanden, wird der aktuelle Fehlercode nach Aufruf von *quit* auf Null gesetzt, als wäre kein Fehler passiert. Besonders wenn die Möglichkeit besteht, daß eine Kommandofolge von anderen Kommandofolgen aufgerufen wird, sollte man im Fehlerfall immer einen Fehlercode mit *quit*

»nach außen geben«, damit eine solche andere Kommandofolge eine Chance hat, auf einen eventuellen Fehler zu reagieren.

10.6 Wie Sie Fehler in Kommandofolgen abfangen

Ein Teil der eben vermittelten Informationen soll nun aber auch in die Praxis umgesetzt werden! Die aktuelle Version von *ZeigeText* meldet zwar korrekt, wenn die auszugebende Datei nicht existiert; andere Fehler können aber passieren, werden nicht erkannt und führen zum Abbruch der Kommandofolge. Die kritische Stelle ist dabei der *copy*-Befehl, der vor der Ausgabe stattfindet. Wenn die Diskette voll oder schreibgeschützt ist, wird er fehlschlagen. Diese Stelle soll in der folgenden Version von *ZeigeText* nun etwas »entschärft« werden.

```
.KEY datei
.DEF datei text
echo "Start von ZeigeText"
failat 25
copy <datei> TO kopie   ; GEFÄHRLICH
IF ERROR
SKIP fehler
ENDIF
failat 10
.
.     Alles OK
type <datei> OPT N
delete kopie
echo "ZeigeText erfolgreich beendet!"
QUIT
.
.     Fehler beim Kopieren
LAB fehler
IF NOT EXISTS <datei>
echo "Fehler: <datei> existiert nicht!"
ELSE
echo "Fehler: Kopieren von <datei> fehlgeschlagen!"
echo "Ist die Diskette voll oder schreibgeschützt ?"
ENDIF
QUIT 10
```

Die einfache, kurze Kommandofolge, mit der wir begonnen haben, ist nun schon recht lang geworden. Sie nutzt fast alle Möglichkeiten, die eine Kommandofolge überhaupt nutzen kann.

Zunächst wird die kritische Stelle, der *copy*-Befehl, mit einem *failat 25* versehen. Selbst wenn der *copy*-Befehl nun fehlschlägt, läuft die Kommandofolge weiter. Unmittelbar hinter *copy* wird nun geprüft, ob der Befehl mißlungen ist und falls ja, zur Marke *Fehler* gesprungen. Falls

kein Fehler auftrat, wird die schon bekannte Befehlsfolge durchlaufen, die Datei am Bildschirm aufgelistet und schließlich mit *quit* die Kommandofolge verlassen. Vorher aber wird der Abbruch-Level mit *failat 10* wieder auf den Normalzustand zurückgesetzt. Das ist in diesem Fall nicht unbedingt nötig. Man sollte es aber stets tun, wenn der Fehlercode zuvor hochgesetzt wurde. Sonst werden spätere Fehler, die man nicht vorhergesehen hat, nämlich einfach ignoriert und nicht gemeldet.

Zur Marke *Fehler* kommt die Kommandofolge überhaupt nur, wenn ein Fehler aufgetreten ist. Es wird dann geprüft, ob die Datei existiert oder nicht. Wenn nicht, wird eine entsprechende Fehlermeldung ausgegeben. Wenn die Datei existiert, ist die Fehlerursache unklar. Dem Benutzer werden dann zwei wahrscheinliche Fehlermöglichkeiten genannnt. Im Fehlerfall wird schließlich noch ein Fehlercode mit *quit* ausgegeben. Dies ist im Moment nicht unbedingt nötig, aber guter Stil, den man sich »für später« angewöhnen sollte.

Ich hoffe, daß Sie anhand dieses letzten großen Beispiels wenigstens schon einen ungefähren Eindruck von den Möglichkeiten bekommen haben, die sich bei der Verwendung von Kommandofolgen auftun. Kurz gesagt, sollten Sie immer dann ins Auge fassen, eine Kommandofolge zu schreiben, wenn Sie im CLI häufiger eine Aufgabe zu erledigen haben, für die Sie mehrere Kommandos benötigen.

Einige Beispiele dazu, wie man sich mit Kommandofolgen das Leben einfacher machen kann, finden Sie im folgenden Kapitel. Dieses Kapitel wird sich unter anderem auch besonders intensiv mit der speziellen Kommandofolge *Startup-Sequence* beschäftigen, die bei jedem Neustart automatisch ausgeführt wird.

11 Tips für CLI-Anwender

In diesem Kapitel werden Sie einige Informationen und Tips finden, die sonst in keines der Kapitel über das CLI hineinpassen. Sie werden sehen, daß im CLI einige Sachen möglich sind, zu denen Sie auf der Workbench einfach nicht fähig sind. Zum großen Teil handelt es sich dabei um Kommandofolgen und besonders um die Modifikation der Kommandofolge Startup-Sequence.

Viele der hier aufgeführten Tips sind nur für fortgeschrittene Anwender gedacht und deshalb mit Vorsicht zu genießen. Aus dem CLI heraus ist es wirklich recht einfach möglich, Disketten unbrauchbar zu machen, während man sich auf der Workbench dafür schon einige Mühe geben muß.

11.1 Die Startup-Sequence

Bei häufiger Anwendung des CLI wäre es doch ganz praktisch, wenn nach dem Start bereits alle Einstellungen, zum Beispiel das aktuelle Dateiverzeichnis, einige logische Laufwerke und so weiter, so gültig wären, wie sie üblicherweise benötigt werden. Gerade für Programmierer, die im CLI mit vielen Dateien jonglieren müssen, würde das einige Handarbeit nach jedem Systemstart sparen. Genau dies ist möglich. Beim Start des Amiga ist nämlich zunächst einmal – vielleicht im Gegensatz zu Ihren bisherigen Annahmen – das CLI aktiv. Es führt dann aber sofort die Kommandofolge Startup-Sequence im Dateiverzeichnis *s* (das gleich dem logischen Laufwerk *S:* ist) auf der Workbench-Diskette aus. Deren letzter Befehl ist ein *endcli*, weshalb man danach »normalerweise« in die Workbench gelangt.

11.1.1 Das Original

Um wie ein Profi mit dem CLI arbeiten zu können, müssen Sie den Inhalt der Startup-Sequence unbedingt kennen – und verstehen. Wenden Sie deshalb einmal Ihr frisch erworbenes Wissen über Kommandofolgen an und betrachten Sie die Datei *S:Startup-Sequence* mit einem Editor.

Sie können dazu einen der folgenden drei Befehle verwenden (der zweite funktioniert natürlich nur dann, wenn Sie *MicroEmacs* in die *Utilities*-Schublade der Workbench-Diskette gelegt haben oder mit *path* für das CLI auffindbar gemacht haben) :

```
1>ed s:startup-sequence
1>microemacs s:startup-sequence
1>notepad s:startup-sequence
```

Sie werden dann in der Startup-Sequence die folgenden Zeilen finden (zumindest in der Version der Workbench-Diskette, die mir zum Test vorlag. Ihre Version dieser Datei sieht vielleicht ein wenig anders aus oder enthält dieselben Befehle in anderer Reihenfolge.):

```
echo "A500 Workbench 1.2 D Version 33.56 23 APR-87"
binddrivers
if EXISTS sys:system
    path s:system add
endif
if EXISTS sys:utilities
    path s:utilitiesadd
endif
dir RAM:
path RAM: add
SetMap d
Addbuffers df0: 20 ; this uses up about 10K of memory ...
LoadWB
failat 30
SetClock >NIL: opt load
Date
endcli > nil:
```

Diese Kommandofolge beginnt mit einer kurzen Meldung, die mit *echo* auf den Bildschirm gebracht wird. Sie haben sie beim Neustart bestimmt schon einmal gelesen. Danach folgt der Befehl *binddrivers*, mit dem die Gerätetreiber, die sich eventuell in der *Expansion*-Schublade befinden, aktiviert werden. Auffällig sind die beiden If-endif-Sequenzen dahinter. Mit diesen wird jeweils geprüft, ob ein bestimmtes Directory auf der Diskette vorhanden ist. Wenn ja, wird es dann dem aktuellen Suchpfad hinzugefügt. Das führt dazu, daß Sie Programme, die in den Directories (Schubladen) *Utilities* und *System* liegen, vom CLI einfach mit ihrem Namen aufrufen können. Sie brauchen dazu also nicht den vollen Pfadnamen anzugeben. Danach wird mit *dir RAM:* dafür gesorgt, daß das RAM-Disk-Piktogramm auf der Workbench sichtbar wird und auch RAM: dem Suchpfad hinzugefügt. *SetMap d* schaltet dann auf die deutsche Tastaturbelegung um (bis dahin arbeiten Sie mit einer US-Tastatur) und der *Addbuffers*-Befehl legt zusätzlichen Pufferspeicher für das interne Laufwerk (df0:) an, der die Arbeit mit Disketten wesentlich beschleunigt. Der Befehl *SetClock* entnimmt dann die aktuelle Uhrzeit der Uhr in der Speichererweiterung Amiga 501 und teilt sie dem Betriebssystem mit. Wenn Sie keine Speichererweiterung in Ihren Amiga 500 eingebaut haben, führt das natürlich zu einer Fehlermeldung, die aber mit >NIL: »ins Leere« gelenkt wird. Der Befehl *failat* direkt vor

SetClock sorgt dafür, daß die Startup-Sequence auch dann nicht abbricht, wenn *SetClock* einen Fehler liefert. Er setzt den Fehlerlevel, ab dem Kommandofolgen abgebrochen werden, auf 30 hoch.

Die beiden entscheidenden Zeilen gegen Ende in dieser Kommandofolge enthalten die Befehle *LoadWB* und *endcli*. *LoadWB* startet die Workbench und erst ab diesem Moment wird diese komfortablere Benutzerschnittstelle überhaupt verfügbar. Danach wird das CLI-Fenster mit *endcli* geschlossen, wobei durch *>nil:* die Abschiedsmeldung des CLI ins »Nichts« geschickt wird.

11.1.2 Modifikationen der Startup-Sequence

Wie Sie sehen, ist das CLI wirklich die »originale« Benutzerschnittstelle des Amiga, die beim Start zunächst einmal aktiv ist. Wenn Sie die beiden Zeilen *loadwb* und *endcli* aus der Startup-Sequence löschen, bleiben Sie deshalb im CLI und kommen erst gar nicht auf die Workbench.

Alternativ können Sie auch nur den Befehl *endcli* löschen, was besonders dann sinnvoll ist, wenn Sie häufig zwischen beiden Möglichkeiten, den Amiga zu bedienen, wechseln wollen. Löschen Sie *endcli*, aber nicht *LoadWB*, haben Sie von Anfang an sowohl die Workbench als auch das CLI zur Verfügung. Das CLI-Fenster verdeckt allerdings zunächst die Workbench. Sie müssen es zuerst verkleinern, um die Disketten-Piktogramme und andere Bestandteile der Workbench zu sehen.

Eine andere – weniger endgültige – Möglichkeit, im CLI zu bleiben, ist das Abbrechen der Kommandofolge während ihrer Ausführung. Jede Kommandofolge – also auch Startup-Sequence – kann mit [Ctrl]+[D] abgebrochen werden. Wenn Sie diese Tastenkombination rechtzeitig drücken, bevor der *endcli*-Befehl ausgeführt wird, bleiben Sie im CLI, ohne Modifikationen an der Startup-Sequence vornehmen zu müssen. Dabei kann es allerdings leicht vorkommen, daß die Startup-Sequence schon abgebrochen wird, bevor der Befehl *setmap d* ausgeführt wird (wodurch Sie eine englische Tastatur bekommen). Und wenn Sie in die Workbench wollen, müssen Sie natürlich selbst *LoadWB* eingeben. Das Abbrechen der Startup-Sequence ist also nicht unbedingt empfehlenswert, da oft schwierig festzustellen ist, welche Befehle schon ausgeführt wurden und welche nicht.

Es gibt eine wesentlich elegantere Möglichkeit, CLI und Workbench gleichzeitig auf den Bildschirm zu bekommen, bei der Sie noch nicht einmal das CLI-Fenster beiseite räumen müssen, um die Workbench-Piktogramme zu sehen. Setzen Sie dazu einfach den folgenden Befehl vor *endcli >NIL:* am Ende der Startup-Sequence:

```
newcli CON:0/200/640/54/CLI_Fenster
```

Dadurch wird ein neues CLI-Fenster erzeugt, das vor dem Schließen des ersten CLI-Fensters erscheint (und am Bildschirm stehenbleibt). Es befindet sich praktischerweise am unteren Bildschirmrand, wo meist keine Piktogramme auftauchen.

11.1.3 Anderweitige Nutzung der Startup-Sequence

Statt kleine Modifikationen der Startup-Sequence vorzunehmen, können Sie die Standardversion natürlich auch ganz löschen und durch eine eigene Version ersetzen, die die Aufgaben durchführt, die Sie üblicherweise immer beim Start des Amiga durchführen.

Falls Sie im CLI bleiben wollen, so können dies zum Beispiel einige *cd-*, *assign-* und *path-*Befehle sein. Für Benutzer des CLI und der Workbench könnte es interessant sein, bei jedem Start *preferences* aufzurufen, um die richtige Uhrzeit und das korrekte Datum einzustellen, wenn Sie keine Speichererweiterung Amiga 501 mit eingebauter Uhr besitzen.

Falls Sie über längere Zeit hinweg immer nur mit einem Werkzeug der Workbench arbeiten, zum Beispiel mit dem Malprogramm Graphicraft oder der Amiga-BASIC, so können Sie dieses Werkzeug auch gleich direkt vom CLI aus starten und brauchen nicht erst auf der Workbench mühselig Disketten- und Schubladen-Piktogramme zu öffnen. Solche Programme sollten Sie aber am besten immer mit Hilfe des *run*-Befehls (zum Beispiel *run df1:AmigaBASIC*) aufrufen, damit die Startup-Sequence nach dem Aufruf des Programms fortfahren kann. Rufen Sie innerhalb der Startup-Sequence ein Programm ohne *run* auf, so wartet das CLI erst auf die Beendigung dieses Programms, bevor es mit den restlichen Befehlen fortfährt.

Dies alles waren aber nur ein paar Vorschläge. Im Laufe der Zeit werden Sie selbst bestimmt noch viele andere nützliche Anwendungen für Startup-Sequence finden. Anregungen dazu werden Sie auch in den folgenden Abschnitten dieses Kapitels noch genug finden.

11.2 Wie Sie auf der Workbench-Diskette Platz schaffen

Das CLI wird aber nicht nur durch Kommandofolgen (wie die Startup-Sequence) interessant. Ein weiterer Vorteil des CLI ist es, daß Sie dort auch Dateien »sehen«, die auf der Workbench verborgen sind. Auf der Workbench-Diskette sind viele dieser Dateien mehr oder weniger überflüssig oder werden nur sehr selten benötigt. Indem Sie sie löschen, können Sie wertvollen Platz auf der Diskette freimachen, den Sie für andere Zwecke einsetzen können. Das ist natürlich besonders für diejenigen Besitzer des Amiga 500 interessant, die noch ein zweites Diskettenlaufwerk und keine Festplatte besitzen.

11.2.1 Inhalt der Workbench-Diskette

Bevor Sie aber an das Löschen »unsichtbarer« Dateien gehen, benötigen Sie einen Überblick darüber, wo welche Dateien auf der Workbench-Diskette zu finden sind und wofür sie benötigt werden. Glücklicherweise ist der Inhalt der Workbench-Diskette recht übersichtlich in Directories aufgeteilt, die die verschiedenen Dateien logisch gruppieren. Die folgende Übersicht stellt den Aufbau des Dateienbaums auf der Workbench-Diskette dar. In ihr sind nur Directories und keine Dateien aufgeführt.

```
c
demos
devs
      clipboards
      printers
      keymaps
Empty
Expansion
Fonts
      diamond
      emerald
      garnet
      opal
      ruby
      sapphire
      topaz
l
libs
s
System
t
Trashcan
Utilities
```

Das Directory *C* enthält – wie Sie ja schon erfahren haben – Programme, die Sie als CLI-Kommandos verwenden können. Viele dieser Kommandos werden Sie nie benötigen. Speziell dann, wenn Sie üblicherweise auf der Workbench arbeiten, benötigen Sie nur die wenigen Programme im Directory *C*, die in der Startup-Sequence vorkommen.

Das Directory *Demo* enthält einige Beispielprogramme, die Punkte, Linien oder Kreise auf den Bildschirm malen. Sie dienen nur zur Demonstration der Multitasking-Fähigkeiten des Amiga 500 und Sie werden diese Programme nach den ersten Wochen Ihres Kontaktes mit dem Amiga wahrscheinlich nie wieder benötigen.

Das Directory *devs* ist ein sehr wichtiges Directory, das Dateien enthält, die Amiga-DOS zur Ansteuerung der verschiedenen Geräte benötigt, aus denen Ihr Amiga-System besteht oder die Sie an den Amiga anschließen. Neben einer Reihe weiterer Directories enthält es die folgenden Dateien:

```
clipboard.device
Mountlist
narrator.device
parallel.device
printer.device
serial.device
system-configuration
```

Die Datei *clipboard.device* wird von einigen Programmen benötigt, um eine Zwischenablage zu realisieren. Lesen Sie die Beschreibung von *Notepad*, wenn Sie mehr über das Prinzip einer Zwischenablage erfahren wollen. Die Dateien *parallel.device* und *perial.device* werden benötigt, um die parallele und serielle Schnittstelle an der Rückseite Ihres Amiga 500 zu steuern. Ähnlich wird *printer.device* zur Steuerung des Druckers benötigt. Und *narrator.device* wird immer dann benötigt, wenn Sie die Sprachsynthese benutzen (zum Beispiel mit dem Programm *Say*). Die Datei Mountlist wurde bereits bei der Beschreibung des Befehls *mount* (im Kapitel 8) erläutert. Und in der Datei system-configuration speichert *Preferences* die Einstellungen (für Workbench-Farben, Mausübersetzung und so weiter), die Sie in *Preferences* gewählt haben.

Das Directory *devs* enthält außer diesen Dateien noch drei weitere Directories. *Clipboards* wird von *clipboard.device* benötigt und braucht Sie nicht weiter zu interessieren (dieses Directory ist fast immer leer). Printers enthält die verschiedenen »Druckertreiber«, die Sie in Preferences auswählen können. Sie benötigen wahrscheinlich nur einen davon, da Sie ja auch nur einen Drucker besitzen. Und das Directory keymaps im Directory devs enthält die verschiedenen Tastaturbelegungen. Auch davon benötigen Sie nur eine (ich nehme einmal an, die deutsche). Damit wäre der Überblick über *devs* und seine Unterdirectories aber auch schon abgeschlossen.

Das Directory *Empty* auf der Workbench-Diskette ist – wie sein Name schon sagt – leer und dient zur Erzeugung neuer Schubladen auf der Workbench.

Auch *Expansion* ist üblicherweise leer, solange Sie keine Hardware-Erweiterungen kaufen, bei denen Sie spezielle Treiberdateien mitbekommen, die dann in dieses Directory gelegt werden müssen.

Fonts hingegen enthält wieder eine Reihe von Dateien und weiteren Directories. Im Directory *Fonts* werden nämlich die verschiedenen Schriftarten abgespeichert, wie Sie wahrscheinlich schon an den Namen der Dateien darin feststellen werden. Zu jeder Schriftart gibt es eine Datei und ein Directory. Beide zusammen werden benötigt, um Schrift in dieser Schriftart darstellen zu können. Wenn Sie zusätzlich Schriftarten erwerben – es gibt inzwischen schon mehrere Disketten voll davon – müssen Sie für jede neue Schriftart eine Datei und ein Directory in das Directory *Fonts* der Workbench-Diskette legen. In vielen Programmen jedoch können Sie auf Schriftarten und damit auf den Inhalt von *Fonts* verzichten.

Das Directory *l* ist hingegen wieder eines der für Amiga-DOS besonders wichtigen Directories. Es enthält drei Dateien. Die Datei *Disk-Validator* wird benötigt, um nach dem Einlegen einer Diskette feststellen zu können, ob sie »in Ordnung« ist. Sie sollten sie nie löschen. Die Datei *Ram-Handler* wird für die RAM-Disk benötigt. Ohne diese Datei haben Sie keine RAM-Disk (Amiga-DOS kennt dann das Laufwerk *RAM:* nicht). Und die Datei *Port-Handler* wird zum Zugriff auf die Schnittstellen (zum Beispiel die serielle und parallele) auf der Rückseite des Amiga 500 benötigt.

Auch das Directory *libs* ist sehr wichtig. Es enthält Bibliotheken mit kleinen Programmen, die von anderen Programmen aus benutzt werden können. Die Namen dieser Bibliotheken enden alle mit »library« (engl. für Bibliothek). Die Datei *diskfont.library* wird benötigt, wenn

Programme die im Directory *Fonts* liegenden Zeichensätze verwenden wollen. Die Bibliotheken *mathtrans* und *mathieeedoubas* enthalten mathematische Routinen (transzendentale Funktionen und mathematische Operationen doppelter Genauigkeit), die nur von wenigen Programmen benötigt werden. Die *icon.library* enthält Routinen zur Darstellung von Piktogrammen und *info.library* wird für Fälle, in denen man mehr zu einer Datei, als nur ihren Namen, wissen will, benötigt. Der Info-Befehl der Workbench benötigt sie zum Beispiel. *Translator.library* wird zur Sprachsynthese benötigt und die *Version.library* teilt Programmen mit, welche Version der Workbench-Sofware sie benutzen.

Das Directory *s* enthält Kommandofolgen. Hier (und im aktuellen Directory) sucht der Befehl *execute* die Kommandofolgen, deren Namen ihm als Parameter angegeben werden.

Das Directory *System* kennen Sie bereits von der Workbench her sehr gut. Es entspricht der Schublade System und enthält (außer den obligatorischen Info-Dateien) auch keine weiteren, unsichtbaren Dateien. *System* enthält unter anderem auch die beiden CLI-Befehle format und diskcopy, die nur deshalb wie die anderen CLI-Kommandos verwendet werden können, weil das Directory *System* in der Startup-Sequence mit dem *path*-Befehl in den Suchpfad für Programme eingehängt wird.

Das Directory *t* ist für temporäre Dateien gedacht. Programme legen in dieses Directory Dateien hinein, die nur vorübergehend benötigt werden (wie zum Beispiel MicroEmacs, das unter Umständen die Datei edit.backup hier anlegt). Auf der originalen Workbench-Diskette ist *t* zunächst einmal leer.

Das Directory *Trashcan* kennen Sie schon von der Workbench her. Es entspricht dem Mülleimer-Piktogramm. Dieses Directory enthält nur dann Dateien, wenn Sie auf der Workbench Piktogramme in den Mülleimer gelegt haben. Diese Dateien werden alle gelöscht, wenn Sie den Mülleimer anklicken und dann den Befehl *Empty Trash* auswählen.

Das letzte Directory der Workbench-Diskette, *Utilities*, enthält zwei zusätzliche Programme, die sowohl vom CLI wie auch von der Workbench aus verwendet werden können. Sie kennen diese beiden Programme aus Kapitel 4. Es sind der Taschenrechner (*Calculator*) und der Notizblock (*Notepad*). Auch andere Programme, die im CLI und auf der Workbench nützlich sein können (wie zum Beispiel *MicroEmacs*) sollten Sie in dieses Directory legen.

Neben diesen eben genannten Directories gibt es natürlich auch noch das Wurzeldirectory der Workbench-Diskette. Dessen Inhalt kennen Sie ebenfalls schon. Es sind neben einer Reihe von Directories die Uhr (*Clock*) und *Preferences*.

11.2.2 Worauf Sie gut verzichten können

Wie Sie im letzten Abschnitt gesehen haben, besitzt die Workbench-Diskette einen recht komplexen Aufbau. Eine Unzahl von Dateien und ineinander verschachtelten Directories tummelt sich darauf. Auf viele davon können Sie aber gut verzichten. Oder anders gesagt: Sie können diese Dateien und Directories löschen. Dazu finden Sie im folgenden einige Tips. Wenn Sie diese Tips ausprobieren wollen, sollten Sie das aber unbedingt auf einer Kopie der

Workbench-Diskette machen, mit der Sie normalerweise arbeiten (die ja auch schon eine Kopie sein sollte).

Achten Sie beim Löschen von Dateien, die man auch auf der Workbench sehen kann, unbedingt darauf, daß Sie immer die Datei zusammen mit der zugehörigen Info-Datei löschen. Löschen Sie nur eine Datei und nicht deren Info-Datei, so ist die gelöschte Datei auf der Workbench scheinbar noch (als Piktogramm) vorhanden. Das kann zu verwirrenden Fehlermeldungen führen. Dateien, die als Piktogramme sichtbar sind, sollten Sie deshalb am besten immer auf der Workbench löschen. Dabei kann ein solches Versäumnis nämlich nicht vorkommen. Mehr zu Info-Dateien erfahren Sie aber auch noch in einem späteren Abschnitt dieses Kapitels.

Nach dieser langen Vorrede aber nun ans Werk: Falls Sie nicht ein intensiver Anwender einer Textverarbeitung sind oder die Möglichkeit zum Wechsel der Schriftart nicht nutzen, sollten Sie zunächst einmal sämtliche Schriftart-Dateien löschen. Das ist recht einfach mit dem folgenden CLI-Befehl:

```
1>delete :fonts/#? all
```

Dieser Befehl löscht alle Dateien und Directories im Directory *Fonts* (ohne dabei das Directory *Fonts* selbst). Das Directory *Fonts* bleibt sicherheitshalber bestehen, weil einige Programme auf seine Existenz Wert legen – selbst dann, wenn sie es leer vorfinden.

Wenn Sie nur einen Zeichensatz löschen wollen – weil er Ihnen zum Beispiel nicht gefällt – können Sie dazu die beiden folgenden Befehle verwenden:

```
1>delete :fonts/<name>.font
1>delete :fonts/<name> all
```

(Dabei müssen Sie <name> natürlich gegen den Namen der Schriftart ersetzen, die Sie löschen wollen.) Aber nicht nur das Schriftarten-Directory eignet sich gut dazu, Platz zu schaffen. Auch aus dem Directory *devs/printers* können Sie alle Dateien bis auf eine löschen. Diese eine Datei ist natürlich die, deren Namen Sie im Programm *Preferences* ausgewählt haben und die zu dem Drucker paßt, den Sie verwenden. Wenn Sie (zumindest mit dieser Workbench-Diskette) überhaupt nicht drucken wollen, können Sie devs/printers mit dem folgenden Befehl komplett löschen:

```
1>delete :devs/printers all
```

Für das Löschen einzelner Dateien eignet sich hingegen der *dir*-Befehl sehr gut. Geben Sie dazu den folgenden Befehl ein:

```
1>dir :devs/printers opt i
```

Sie bekommen dann die Dateien im Directory *devs/printers* einzeln aufgelistet. Nach jedem Namen haben Sie die Möglichkeit, die Datei zu löschen, indem Sie »del« und [Return] eingeben oder sich mit [Return] den nächsten Namen zeigen lassen.

Diese Option des *dir*-befehls eignet sich natürlich nicht nur zum selektiven Löschen von Dateien im *printers*-Directory. Sie können damit natürlich auch ganze Disketten nach Löschkandidaten »durchkämmen«. Lesen Sie dazu im Zweifelsfall noch einmal die Beschreibung des *dir*-Befehls im Kapitel 8 nach.

Nun aber zurück zum Thema *Drucken*. Wenn Sie gar nicht drucken wollen, können Sie im Directory *devs* außer dem Directory *printers* auch noch die Dateien, *serial.device, parallel.device* und *printer.device* löschen. Dann können Sie allerdings auch keine anderen Geräte an die serielle oder parallele Schnittstelle auf der Rückseite des Amiga 500 anschließen. Und wenn Sie gerade sowieso im Directory *devs* »zugange sind«, können Sie auch gleich noch die Dateien *narrator.device* und *Mountlist* löschen. *Narrator.device* benötigen Sie nur, wenn Sie Sprachausgabefähigkeiten des Amiga nutzen wollen und *Mountlist* nur dann, wenn Sie ein externes Disketten- oder Festplattenlaufwerk angeschlossen haben, das sich nicht exakt so verhält wie ein 3,5-Zoll-Diskettenlaufwerk.

Viele Möglichkeiten, Platz zu schaffen, bietet auch das Directory *c*, das die CLI-Kommandos enthält. Viele dieser Kommandos benötigen Sie wahrscheinlich nie oder nur äußerst selten. Sie können in diesem Directory deshalb nach Herzenslust löschen – besonders dann, wenn Sie eigentlich ein typischer Workbench-Anwender sind oder sich eine Diskette schaffen wollen, die nur für das Arbeiten mit der Workbench gedacht ist. (Eine vollständige Version der Workbench-Diskette haben Sie ja sowieso aufgehoben – oder?) Beachten Sie aber bitte, daß einige der Befehle im *c*-Directory auch dann benötigt werden, wenn Sie selber nie das CLI verwenden. Damit die Startup-Sequence fehlerfrei ablaufen kann, müssen die in ihr verwendeten Kommandos natürlich verfügbar sein. Dazu gehören:

```
echo
binddrivers
if
endif
dir
path
setmap
addbuffers
failat
setclock
date
loadwb
endcli
```

Ihre Version der Workbench-Diskette kann unter Umständen aber eine Startup-Sequence mit noch mehr anderen Kommandos enthalten. Lassen Sie sich die Startup-Sequence deshalb am besten mit *type s:startup-sequence* anzeigen und notieren Sie sich alle darin vorkommenden CLI-Kommandos. Verwenden Sie dann am besten wieder den *dir*-Befehl mit der Option *opt i* um Dateien im *c*-Directory zu löschen. Legen Sie dieses Buch dazu neben Ihre Tastatur und schlagen Sie Kapitel 8 auf. Löschen Sie dann die Befehle, von denen Sie meinen, daß Sie

darauf verzichten können. Lesen Sie die Funktion der Befehle im Zweifelsfall im Kapitel 8 nach. Und löschen Sie vor allem keine Befehle, die in der Startup-Sequence vorkommen! (CLI-Profis können natürlich auch einige Zeilen aus der Startup-Sequence löschen und dann die entsprechenden Befehle im *c*-Directory entfernen. Dazu sollten Sie aber wirklich sicher sein, auf diese Zeilen verzichten zu können.)

Nach dieser riskanten Operation nun wieder eine einfache. Löschen Sie das Directory Demos samt Inhalt. Auf die kleinen Beispielprogramme darin können Sie sicherlich verzichten. Dazu genügt der folgende Befehl:

```
1>delete :demos all
```

Genauso können Sie wahrscheinlich auf einen Teil der Programme in den Directories Utilities, System und im Wurzeldirectory der Diskette verzichten. Gehen Sie zum Löschen in diesen Directories aber am besten auf die Workbench. Es ist auf der Workbench wesentlich leichter, die Programme und Directories zusammen mit Ihren Piktogrammen zu löschen. Klicken Sie sie einfach nacheinander an, wobei Sie die [Shift]-Taste gedrückt halten und wählen Sie dann *Discard* aus dem Menü *Workbench*.

Im Directory *System* können Sie zum Beispiel relativ gefahrlos alle Programme bis auf *CLI*, *Format* und *DiskCopy* löschen. Und diese Programme benötigen Sie auch nur dann, wenn Sie von der Workbench in das CLI wollen, wenn Sie Disketten formatieren oder Disketten kopieren wollen. Falls Sie (zumindest, wenn Sie mit dieser Diskette starten) auf diese drei Arbeiten verzichten können, sind auch *CLI*, *Format* und *DiskCopy* überflüssig. Die Schublade (Directory) Utilities können Sie komplett löschen, wenn Sie auf *Notepad* und *Calculator* verzichten können. Als CLI-Anwender dürfte Ihnen dieser Verzicht im allgemeinen nicht schwerfallen. Und im Wurzeldirectory der Workbench-Diskette können Sie recht gefahrlos *Clock* und *Preferences* löschen. Bedenken Sie aber bitte, daß Sie ohne Preferences unter anderem den Mauszeiger und die Workbenchfarben nicht mehr ändern können und keinen anderen Drucker mehr wählen können.

Kommen wir nun zu den beiden Directories *l* und *libs*. In beiden ist das Löschen von Dateien etwas riskant. Lesen Sie die folgenden Erläuterungen deshalb gut durch! Im Directory *l* können Sie unter Umständen die Dateien *Port-Handler* und *Ram-Handler* löschen; *Disk-Validator* sollten Sie auf alle Fälle unangetastet lassen. Wenn Sie *Ram-Handler* löschen, verfügen Sie allerdings über keine RAM-Disk mehr und wenn Sie *Port-Handler* löschen, können Sie nicht mehr auf die Schnittstellen an der Rückseite des Amiga 500 zurückgreifen (in diesem Fall sollten Sie auch *serial.device*, *parallel.device* und *printer.device* löschen, wie oben beschrieben).

Die einigermaßen risikolosen Fälle im *libs*-Directory sind *version.library* und *translator.library*. Die Datei *version.library* wird in fast keinem Programm benötigt und *translator.library* nur zur Sprachsynthese. Wenn Sie also darauf verzichten können, Ihren Amiga sprechen zu hören; weg mit der *translator.library*, die recht groß ist.

11.2.3 Die Mini-Workbench für Sparsame

Nachdem Sie nun erfahren haben, auf welche Dateien auf der Workbench-Diskette Sie »unter Umständen« verzichten können, möchte ich Ihnen nun eine Art Mini-Workbench vorstellen. Diese Diskette enthält nur das Nötigste für die Arbeit im CLI und auf der Workbench und enthält nicht einmal 190 Kbyte Daten. Fast 700 Kbyte sind frei. Die folgende Liste zeigt Ihnen einen Überblick über den Inhalt dieser Diskette, ähnlich wie sie der Befehl *dir opt a* ergeben würde

```
Trashcan (dir)
c (dir)
     assign              cd
     copy                delete
     dir                 echo
     endcli              execute
     loadwb              newcli
     path                run
     setclock            setmap
l (dir)
     Ram-Handler         Disk-Validator
devs (dir)
     keymaps (dir)
        d
     clipboards (dir)
     clipboard.device    System-configuration
s (dir)
     Startup-Sequence
t (dir)
fonts (dir)
libs (dir)
     diskfont.library    icon.library
     info.library        version.library
.info                    Disk.info
Notepad                  Notepad.info
Trashcan.info
```

Diese Diskette enthält sicherlich noch nicht das absolute Minimum an Dateien. Für meine eigenen Zwecke ist sie aber ideal und ermöglicht ein Arbeiten sowohl im CLI wie auch auf der Workbench. Wo das absolute Minimum an Dateien liegt, ist auch schwer zu sagen. Es hängt von den Programmen ab, die Sie verwenden. Verschiedene Programme machen nämlich verschiedene Annahmen über die Dateien und Directories, die Sie im logischen Laufwerk SYS: (also auf der Workbench-Diskette) erwarten. Die meisten von mir selbst benutzten Programme (zum Beispiel Notepad und Deluxe Paint II) arbeiteten einwandfrei, wenn ich meinen Amiga 500 mit dieser Diskette startet. Es ist aber nicht ausgeschlossen, daß anderen Programmen gewisse Dateien fehlen (zum Beispiel die mathematischen Bibliotheken im

libs-Directory). Auch ist mit dieser Diskette keine Sprachausgabe möglich, weil die entsprechenden Bibliotheken darauf fehlen. Bevor Sie selbst damit beginnen, ständig mit einer solchen Mini-Workbench zu arbeiten, sollten Sie die Programme, mit denen Sie später arbeiten wollen, ausführlich testen, nachdem Sie einen Neustart mit dieser Workbench-Diskette durchgeführt haben.

Eine so sparsam gefüllte Diskette können Sie auf zwei verschiedene Arten erstellen. Zum einen können Sie die normale Workbench-Diskette kopieren und dann auf der Kopie alle überflüssigen Dateien und Directories löschen. Zum anderen können Sie aber auch neue, leere Disketten nehmen, sie formatieren, mit *install* startfähig machen und dann die gewünschten Dateien von einer vollständigen Workbench-Diskette hinüberkopieren. Die zweite Methode ist allerdings nur für Besitzer eines zweiten Diskettenlaufwerks praktikabel. Mit nur einem Laufwerk müssen Sie dazu sonst ein unerträgliches Diskettenwechseln beginnen.

Diese Mini-Workbench ist aber kein Selbstzweck. Im täglichen Einsatz werden Sie die knapp 700 Kbyte, die darauf noch frei sind, wahrscheinlich nicht frei lassen. Statt dessen können Sie Programme und Daten auf diese Diskette legen und so mit einer Diskette auskommen, wo Sie sonst zwei benötigen würden. Für Besitzer nur eines Diskettenlaufwerks (des internen) bedeutet das weniger Diskettenwechsel und für Besitzer von zwei Laufwerken wird ein Laufwerk frei für zusätzliche Datendisketten.

Fertigen Sie deshalb sofort dann, wenn Sie sich eine Mini-Workbench zusammengelöscht haben, eine oder mehrere Kopie(n) dieser Diskette an und legen Sie das Original beiseite. Auf diese Kopien können Sie nun gut ein oder zwei Programme legen und müßten dann noch genug Platz für ein paar Projekte (Daten) haben. (Dies ist bei kopiergeschützten Programmen leider schwierig oder unmöglich.)

Auch für Programmierer ist eine solche Diskette sehr interessant. Um zum Beispiel mit Amiga-BASIC zu arbeiten, benötigen Sie normalerweise zwei Disketten (und damit idealerweise auch zwei Diskettenlaufwerke). Wenn Sie das Piktogramm Amiga-BASIC aber auf die Mini-Workbench-Diskette legen, haben Sie noch reichlich Platz für eigene Programme und Hilfsdateien.

11.3 Die RAM-Disk

Eines der nützlichsten Hilfsmittel des CLI ist die RAM-Disk. Diese Diskette, die stets den Namen RAM: trägt, wird ja im RAM-Speicher des Amiga simuliert und ist deshalb viel schneller als es jede »echte« Diskette oder Festplatte sein kann. Zudem haben Sie mit der RAM-Disk ein zusätzliches Diskettenlaufwerk zur Verfügung. Wenn alle Disketten, die sich in den wirklichen Laufwerken befinden, schon voll sind, bietet die RAM-Disk oft willkommenen Ausweichplatz. Eine (vorsichtige) Anwendung der RAM-Disk kann Ihnen deshalb viel Zeit und Mühen ersparen. Das gilt natürlich nicht nur im CLI, sondern genauso für die Workbench. Im CLI läßt sich die RAM-Disk jedoch viel einfacher nutzen.

11.3.1 Programme und Dateien in der RAM-Disk

Jedes Programm, das in die RAM-Disk kopiert wurde, kann ungleich schneller gestartet werden. Auch die Kommandos des CLI sind ja Programme. Jedesmal, wenn Sie ein solches Kommando aufrufen, muß dieses Programm erst von der Diskette geladen werden. Deshalb dauert es oft einen kleinen Moment, bis das CLI auf einen Befehl reagiert. Eine mögliche Nutzung der RAM-Disk wäre es deshalb, einen Teil der am häufigsten benutzten CLI-Befehle auf die RAM-Disk zu kopieren und dann das logische Laufwerk C: »umzulegen«. Die nachstehende Folge von CLI-Befehlen (aus der man natürlich eine Kommandofolge machen könnte) leistet genau das:

```
makedir ram:c
copy :c/copy ram:c/copy   ; copy ist nun schneller
path ram:  add            ; ab jetzt wird copy im RAM
verwendet
copy :c/dir     ram:c
copy :c/cd      ram:c
copy :c/delete  ram:c
copy :c/list    ram:c
copy :c/assign  ram:c
copy :c/execute ram:c
copy :c/run     ram:c
......                 ; und so weiter für alle wichtigen CLI-Befehle
```

Wichtig ist bei dieser Kommandofolge der *path*-Befehl, der dem CLI sagt, wo die Kommandos zu finden sind. Würde er wegfallen, lägen die ausgesuchten Kommandos zwar auf der RAM-Disk, das CLI würde aber nach wie vor die Kommandos auf der Diskette benutzen. Wenn Sie aber ein CLI-Kommando eingeben, nachdem Sie diese Folge von Kommandos eingegeben haben, sucht das CLI zunächst im aktuellen Directory nach dem entsprechenden Programm dann in den anderen Directories des Suchpfades und schließlich auf der RAM-Disk. Sie brauchen aber nicht alle CLI-Kommandos auf die RAM-Disk zu legen. Wird dort nämlich ein (selten benötigtes) Kommando nicht gefunden, sucht danach das CLI dieses Programm im logischen Laufwerk *C*. Und dieses ist (sofern Sie diese Zuordnung nicht mit *assign* geändert haben) gleich dem Directory *c* der Workbench-Diskette, die ja den vollen Satz von CLI-Kommandos enthält.

Wenn Sie häufiger eine Gruppe von CLI-Kommandos in der RAM-Disk unterbringen wollen, sollten Sie sich die oben vorgestellte Befehlsfolge in eine Kommandofolge im Directory *s* legen. Sie können diese Befehlsfolge dann mit einem einzigen CLI-Befehl ausführen. Die folgende Kommandofolge leistet genau das und kann gleichzeitig auch dazu verwendet werden, die RAM-Disk bei Bedarf wieder zu leeren:

```
IF EXISTS RAM:c          ; Kommandos schon im RAM
delete RAM:c all
ELSE                     ; Kommandos noch nicht im RAM
```

```
makedir ram:c
copy :c/copy ram:c/copy ; copy ist nun im RAM
path ram:c  add         ; ab jetzt wird copy im RAM verwendet
copy :c/dir     ram:c
copy :c/cd      ram:c
copy :c/delete  ram:c
copy :c/list    ram:c
copy :c/assign  ram:c
copy :c/execute ram:c
copy :c/run     ram:c
ENDIF
```

Diese Kommandofolge überprüft zunächst, ob es das Directory *c* auf der RAM-Disk schon gibt. Wenn ja, wird es gelöscht, wenn nein, wird es mit *makedir* angelegt und dann die gewünschten Kommandos hineinkopiert. Die Liste der Kommandos, die kopiert werden, könnte bei Ihnen natürlich länger (oder auch kürzer) sein.

11.3.2 Kombination von RAM-Disk und Startup-Sequence

Falls Sie eine bestimmte Gruppe von CLI-Kommandos immer in der RAM-Disk haben wollen, können Sie sich die oben gezeigte Befehlsfolge natürlich auch an das Ende der Startup-Sequence legen. Der Startvorgang dauert dann zwar immer etwas länger. Dafür geht aber danach alles viel schneller.

Besonders für Programmierer ist die Kombination RAM-Disk und Startup-Sequence noch in einer weiteren Hinsicht interessant. Viele Compiler-Sprachen benötigen bei der Übersetzung eines Programms eine ganze Reihe von Hilfs-Dateien (bei der Programmiersprache C zum Beispiel die .h-Dateien und die *.lib*-Bibliotheken). Wenn man diese Dateien schon beim Start in die RAM-Disk legt, so führt das zu einem wesentlich beschleunigten Übersetzungsvorgang – und den weiß jeder Programmierer zu schätzen. (Auch die temporären Dateien, die von einem Compiler meist während der Übersetzung angelegt werden, sollte man, sofern das möglich ist, auf die RAM-Disk schicken. Hierzu müssen Sie meist nur mit dem *assign*-Befehl ein bestimmtes logisches Laufwerk auf die RAM-Disk umlegen.)

Das Kopieren von Dateien in die RAM-Disk sollten Sie aber – gerade von der Startup-Sequence aus – auch nicht übertreiben. Das dabei auftretende Problem ist naheliegend: sie verschlingt kostbaren Speicherplatz, den man vielleicht anderweitig benötigen würde. So werden viele Programme zwar schneller, wenn sie Dateien auf der RAM-Disk lesen und schreiben. Dafür kann es aber sein, daß nicht mehr so viele Programme gleichzeitig in den Speicher passen. Im Extremfall kann man, wie auf den meisten anderen Computern, nur noch mit einem Programm gleichzeitig arbeiten. Will man nun von einem Programm zum anderen wechseln, so muß man erst das eine beenden und das andere starten. Diese Vorgänge können sehr langwierig sein und einen großen Teil der durch die RAM-Disk gewonnenen Geschwindigkeit wieder wettmachen.

11.3.3 Die RAM-Workbench

Zum Abschluß dieses Abschnitts möchte ich Ihnen aber noch einen Extremfall der RAM-Disk-Nutzung vorstellen, der nur zusammen mit der Speichererweiterung Amiga 501 sinnvoll ist. Sie können die RAM-Disk nämlich auch zur Workbench-Diskette »erklären«. Hierzu müssen Sie zunächst einmal alle Directories und Dateien auf die RAM-Disk legen, die es auf einer Workbench-Diskette geben muß. Danach müssen Sie mit dem *assign*-Befehl die verschiedenen logischen Laufwerke, die von Amiga-DOS benutzt werden, auf die RAM-Disk »umlegen«. Falls es Ihnen Schwierigkeiten bereiten sollte, herauszufinden, welche »Directories und Dateien es auf einer Workbench-Diskette geben muß«, so erinnern Sie sich bitte an den Abschnitt »Die Mini-Workbench« weiter oben in diesem Kapitel. Dort wurde eine Art minimale Workbench-Diskette beschrieben.

Wenn Sie in das Directory *s* dieser Mini-Workbench-Diskette die folgende Kommandofolge legen, können Sie mit einem einzigen CLI-Befehl die RAM-Disk zur Workbench-Diskette machen:

```
copy   SYS:   RAM: all
assign SYS:   RAM:
assign S:     RAM:s
assign L:     RAM:l
assign C:     RAM:c
assign FONTS: RAM:fonts
assign DEVS:  RAM:devs
assign LIBS:  RAM:libs
cd RAM:
echo "Sie können die WB_Diskette nun herausnehmen"
```

Starten Sie dazu einfach Ihren Lieblingseditor und geben Sie diese Kommandofolge ein. Sichern Sie sie dann unter dem Namen »S:RamWB«. Sie können sie dann jederzeit mit dem Befehl *execute ramwb* aufrufen. Sabald die Abschlußmeldung (*Sie können die WB_Diskette nun herausnehmen*) am Bildschirm erscheint, ist die RAM-Disk zur Workbench-Diskette geworden. Falls nicht andere Directories der echten Workbench-Diskette im Suchpfad liegen, können Sie diese nun aus dem Laufwerk nehmen und brauchen Sie nie wieder.

11.4 Wie Sie zusätzliche Drucker und Schriftarten verwenden

Wie Sie gerade gesehen haben, ist eine optimale Nutzung der RAM-Disk fast nur vom CLI aus möglich. Auch andere Aufgaben lassen sich ausschließlich im CLI erledigen. Dazu gehört auch die Installation eines Druckers, der nicht schon auf der Standard-Workbench-Diskette »vorgesehen ist«.

Das Programm *Preferences* bietet Ihnen ja im Druckerauswahl-Fenster, in das Sie gelangen, wenn Sie im Hauptfenster den Knopf *Change Printer* anklicken, eine Liste von Druckern an, in der Sie Ihren Drucker auswählen können. Als CLI-Benutzer wissen Sie natürlich

inzwischen, daß es sich dabei um die Liste der Druckertreiber-Dateien im Directory *devs/printers* auf der Workbench-Diskette handelt. Wenn Sie aber einen Drucker besitzen, der nicht in dieser Liste vorkommt und auch zu keinem Drucker in der Liste kompatibel ist, hilft Ihnen die schöne Auswahlmöglichkeit überhaupt nichts.

11.4.1 Wie Sie einen Druckertreiber installieren

Falls Sie aber einen Druckertreiber für Ihren Drucker auftreiben können, ist das Problem nur noch halb so groß. Dann brauchen Sie diesen nämlich nur noch in das Directory *devs/printers* auf der Workbench-Diskette zu kopieren. Sie können das sofort am Beispiel ausprobieren. Auf der Extras-Diskette befinden sich nämlich einige zusätzliche Druckertreiber, die aus Platzgründen nicht mit auf die Workbench-Diskette gelegt wurden. Einer davon ist für den Drucker XEROX 4020 gedacht. Wenn Sie diesen Treiber benutzen wollen (sie also einen XEROX 4020 an Ihren Amiga 500 angeschlossen haben) brauchen Sie nur den folgenden Befehl im CLI einzugeben:

```
1>copy ExtrasD:devs/printers/XEROX_4020  SYS:devs/printers
```

Wenn Sie keine zwei Laufwerke besitzen oder die Extras-Diskette nicht eingelegt haben, fordert Sie das CLI dabei allerdings zu einem oder mehreren Diskettenwechseln auf. Falls Sie sowohl die Extras- wie auch die Workbench-Diskette schon eingelegt hatten, brauchen Sie keine Diskettenwechsel durchzuführen.

Wenn Sie nun Preferences wieder starten, bekommen Sie in der Liste oben rechts im Druckerauswahlfenster XEROX_4020 angezeigt (wozu Sie eventuell die Rollpfeile betätigen müssen) und können auch diesen Druckertreiber verwenden.

11.4.2 Wie Sie zusätzliche Schriftarten installieren

Genauso einfach ist es, weitere Schriftarten zusätzlich zu den schon auf der Workbench-Diskette vorhandenen zu verwenden. Programme, die verschiedene Schriftarten verwenden können (wie zum Beispiel *Notepad*), suchen nach diesen Schriften im logischen Laufwerk *FONTS:* (das beim Start zunächst dem Directory *Fonts* zugeordnet wird). Sie können nun auf zwei verschiedene Arten erreichen, daß das *Notepad* Ihnen eine andere Auswahl von Schriftarten bietet.

Wenn Sie mit dem assign-Befehl vor dem Start von *Notepad* das logische Laufwerk *FONTS:* umdirigieren, sucht *Notepad* an anderer Stelle nach den Schriftarten. Angenommen, Sie besitzen eine Diskette namens *Schrift*, die in einem Directory namens *KlassischeSchriften* eine Auswahl verschiedener Schriftarten enthält. Dann können Sie mit dem folgenden Befehl erreichen, daß Ihnen diese Schriften im *Notepad* zur Verfügung stehen (aber nicht mehr die »normalen« von der Workbench-Diskette).

```
1>assign FONTS: Schrift:KlassischeSchriften
```

Wenn Sie aber wollen, daß Ihnen diese Schriften immer und zusätzlich zu den normalen Schriften zur Verfügung stehen sollen, müssen Sie die Schriftart-Dateien in das Directory *Fonts* auf der Workbench-Diskette kopieren. Wenn das Directory *KlassischeSchriften* aus dem

obigen Beispiel nur Schriftarten enthält, können Sie diese mit dem folgenden Befehl der normalen Auswahl von Schriften hinzufügen:

```
1>copy Schrift:KlassischeSchriften SYS:fonts all
```

Wenn Sie einzelne Schriften kopieren wollen, beachten Sie bitte, daß eine Schriftart immer zwei Bestandteile hat. Im *Fonts*-Directory ist die Schrift *diamond* zum Beispiel durch die Datei *diamond.font* und durch das Directory *diamond* vertreten. Jede Datei im Directory *diamond* entspricht einer Größe der Schrift. Wollen Sie diese Schrift auf einer anderen Diskette kopieren, müssen Sie sowohl die Datei wie auch das komplette Directory (samt den darin enthaltenen Dateien) kopieren. Das könnten Sie zum Beispiel mit den folgenden CLI-Befehlen tun (wobei <Ziel> der Pfadname des Directory ist, in das die Schriftart kopiert werden soll):

```
1>makedir <Ziel>/diamond
1>copy SYS:fonts/diamond <Ziel>/diamond all
1>copy SYS:fonts/diamond.font <Ziel>
```

11.5 Piktogramme vom CLI aus betrachtet

Zum Abschluß dieser kleinen Sammlung von Tips und Tricks möchte ich mich noch etwas um die Verbindung zwischen CLI und Workbench verdient machen. Bis jetzt waren die Tips ja zu einem großen Teil für Anwender gedacht, die vermehrt im CLI tätig werden wollen. Nun werde ich Ihnen zeigen, wie man eine Brücke vom CLI zur Workbench schlagen und zum Beispiel »unsichtbare« Dateien sichtbar machen kann.

11.5.1 Was es mit den Info-Dateien auf sich hat

In diesem und den vorangegangenen Kapiteln sind ja schon mehrfach die Info-Dateien angesprochen worden. Diese tragen denselben Namen wie jeweils eine andere Datei – nur mit der zusätzlichen Endung ».info«. Auch Disketten besitzen eine solche Info-Datei. Diese heißt aber immer Disk.Info und liegt im Wurzeldirectory der Diskette. Solche Info-Dateien sind hauptsächlich für die Workbench gedacht. Sie enthalten unter anderem das Piktogramm, das auf der Workbench die entsprechende Datei oder das Directory repräsentiert, die Position, an der dieses Piktogramm innerhalb des Schubladen-Fensters dargestellt wird und alle anderen Informationen, die Sie mit dem *Info*-Befehl aus dem *Project*-Menü betrachten und ändern können.

Für Programmierer steht eine Bibliothek zur Verfügung (*info.library*), mit deren Hilfe diese Informationen zu einer beliebigen Datei ermittelt werden können, ohne daß man dazu das Format der Info-Datei kennen muß. Nur wenige Programme benutzen aber im Moment diese Informationen überhaupt – hauptsächlich dienen sie zu Ihrer Information und dazu, eine Verbindung zwischen Programmen und ihren Projekten herzustellen.

11.5.2 Wie man eine Datei sichtbar macht

Wenn Sie aber gar keine komplizierten Operationen mit einer Info-Datei vorhaben, sondern einfach nur eine Datei (oder ein Dateiverzeichnis) sichtbar machen wollen, die Sie zwar mit dem *dir*-Befehl im CLI, nicht aber auf der Workbench als Piktogramm sehen, ist das verhältnismäßig einfach und ohne großen Programmieraufwand möglich.

Dazu brauchen Sie dieser Datei nur eine entsprechende Info-Datei zu verschaffen. Der einfachste Weg, eine solche Datei zu erzeugen, ist das Kopieren einer anderen Info-Datei. Der folgende Befehl macht zum Beispiel eine Datei namens ed (die Ihnen bekannt vorkommen sollte), die sich im Dateiverzeichnis *c* befindet (fast) sichtbar.

```
1>copy :utilities/Notepad.info :c/ed.info
```

Der »Haken« an diesem Befehl ist nur, daß damit zwar *ed* schon ein Piktogramm hat, aber in einem Directory liegt, das nicht über ein Piktogramm verfügt. Auf der Workbench gilt ja immer die Regel, daß Sie eine Schublade erst öffnen müssen, bevor Sie die darin befindlichen Objekte sehen. Und die Schublade (das Directory) *c* können Sie nicht öffnen, da es kein Piktogramm hat. Aber auch dieser Zustand läßt sich relativ schnell beheben, indem Sie den folgenden Befehl im CLI-Fenster eingeben:

```
1>copy :utilities.info :c.info
```

Dadurch wird die Schublade *c* sichtbar. Sie können sich davon überzeugen, indem Sie auf die Workbench gehen, das Diskettenfenster der Workbench-Diskette gegebenenfalls schließen und wieder öffnen. An der Stelle, an der zuvor nur die Schublade Utilities lag, liegen nun zwei Schubladen, *Utilities* und *c*, übereinander. Ergreifen Sie die oberste davon mit der Maus und legen Sie sie etwas zur Seite. Sie können nun beide Schubladen in voller Pracht und Schönheit sehen. Öffnen Sie nun *c* mit einem Doppelklick und nach geraumer Zeit, in der das Diskettenlaufwerk heftig arbeitet, erscheint ein Piktogramm namens *ed*. Von der Vielzahl der Programme im Directory *c* haben Sie damit also eines, *ed*, sichtbar gemacht.

Alle anderen Kommandos sind noch unsichtbar (aber Sie wissen jetzt ja, wie Sie sie sichtbar machen können). Es wäre aber zum Beispiel schon möglich, fast alle CLI-Kommandos auf einen Schlag zu löschen, indem Sie die Schublade *c* in den Mülleimer werfen und diesen dann leeren. Davor sollten Sie sich aber hüten – denn dann haben Sie keine CLI-Kommandos mehr und können diese Diskette dann auch nicht mehr als Workbench-Diskette verwenden!

Versuchen Sie aber bitte nicht, *ed* zu starten, indem Sie einen Doppelklick auf sein Piktogramm machen. Der CLI-Editor *ed* hat es nicht gerne, wenn er von der Workbench aufgerufen wird und reagiert darauf nur mit einem Programmabsturz, der von einem System-Requester dokumentiert wird.

11.5.3 IconEd

Mit der oben vorgestellten Methode können unsichtbare Dateien und Schubladen wenigstens schon einmal sichtbar gemacht werden. Die Auswahl für die Piktogramme, mit denen zuvor unsichtbare Dateien auf der Workbench auftauchen, ist jedoch beschränkt – auf genau die

Icons, die schon in irgendeiner Schublade zu finden sind. Für den, der eigene Piktogramme erstellen will, gibt es den Piktogramm-Editor *IconEd*, der bereits in Kapitel 5 ausführlich beschrieben wurde.

Auch mit *IconEd* können Sie übrigens Dateien und Directories sichtbar machen. Dazu müssen Sie nur das Piktogramm eines sichtbaren Objekts laden und dann unter dem Namen eines noch unsichtbaren Objekts (einer Datei oder eines Piktogramms) wieder abspeichern. Dieser Vorgang ist genau betrachtet auch nichts anderes als das Kopieren einer Info-Datei.

Sie müssen beim Kopieren einer Info-Datei – egal ob mit dem *copy*-Befehl oder mit *IconEd* – aber immer darauf achten, daß Sie Piktogramme der richtigen Art kopieren. Wenn Sie einer Schublade ein Piktogramm verschaffen wollen, müssen Sie dazu ein Schubladen-Piktogramm kopieren. Und für Programme und andere (nicht lauffähige) Dateien müssen Sie Piktogramme von Werkzeugen beziehungsweise Projekten kopieren. Wenn Sie diese Regel nicht beherzigen, können bei einem Doppelklick auf die neuen Piktogramme die merkwürdigsten Effekte und Fehlermeldungen auftauchen.

12 Amiga 500, das Grafikwunder

Mit diesem Kapitel beginnt der dritte Teil des vorliegenden Buchs. Während Sie in den vorangegangenen beiden Buchteilen hauptsächlich eine Anleitung zur praktischen Arbeit mit dem Amiga 500 auf der Workbench und im CLI gefunden haben, dienen die restlichen Kapitel mehr dazu, Ihren Wissensdurst zu stillen. Sie werden zum Beispiel in diesem Kapitel einiges über die Grafik-Hardware und -Software des Amiga 500 erfahren. Leider konnte ich mich aber nicht dazu durchringen, zu diesem Thema gleich Bauanleitungen beizulegen. Wer also nach Schaltplänen und fertigen Maschinencodeprogrammen sucht, wird das – zumindest in diesem Buch – vergeblich tun. Wer jedoch mehr darüber erfahren möchte, was den Amiga 500 zu einem so ungemein leistungsfähigen und schnellen Computer macht, oder was noch alles »in ihm steckt«, das die bisher vorgestellte Software nicht demonstrieren konnte, der möge weiterlesen. Ich habe mir Mühe gegeben Fachjargon weitgehend zu vermeiden, so daß vielleicht auch der »Computer-Anfänger« diesem und den folgenden Kapiteln das eine oder andere Interessante über die Innereien seines Amiga 500 entnehmen kann. Zunächst einmal geht es in diesem Kapitel um die grafischen Fähigkeiten des Amiga 500 – sowie natürlich auch seiner Brüder beziehungsweise Schwestern Amiga 1000 und Amiga 2000.

12.1 Grundlegende Einschränkungen der Computergrafik

Um zu verstehen, wozu der Amiga 500 in grafischer Hinsicht fähig ist, benötigt man zunächst einmal ein paar Informationen über die Grundlagen der Computergrafik »an sich« und über einige grundsätzliche Eigenschaften eines Computers.

12.1.1 Das digitale Prinzip

Zu diesen Eigenschaften gehört unter anderem, daß ein Computer alles in kleinen Schritten, man sagt dazu auch »diskret« oder »digital«, macht. Es gibt keine fließenden Bewegungen, sondern immer nur kleine Sprünge, keine wirklich runden Kreise, sondern nur Vielecke. Jeden

Zustand, den ein Programm berücksichtigt oder zeigt, speichert der Computer intern als eine ganze Zahl ab.

Digitaluhren sind ein gutes Beispiel für dieses digitale Prinzip. Sie zeigen vielleicht Stunden, Minuten und Sekunden an, kennen aber keine feinere Unterteilung (von sportlichen Uhren mit Zehntel- oder -Hundertstelsekunden-Teilung einmal abgesehen). Für die Sekunden ist intern in diesen Uhren ein Teil in der Schaltung der Uhr vorgesehen, der genau 60 verschiedene Zustände annehmen kann, beziehungsweise 60 verschiedene Zahlen speichern kann. Eine solche Schaltung kennt keine Zehntelsekunden und keine Zahlen zwischen 59 und 60. Sie bewegt sich nicht langsam von der 59sten auf die 60ste Sekunde zu, sondern mit einem Sprung – genau wie es die Sekundenziffern der Uhr auch anzeigen.

12.1.2 Digitale Farben

Solche Auswirkungen des digitalen Prinzips, Sprünge also, gibt es auch bei der Computergrafik. Während es in der Natur zum Beispiel nahezu unendlich viele Farben gibt (von denen das menschliche Auge aber nur endlich viele unterscheiden kann), steht einem Computer immer nur eine ganz bestimmte Anzahl von Farben zur Verfügung.

Der Amiga kann zum Beispiel etwas mehr als 4000 (genau 4096) verschiedene Farben darstellen. Er kann nicht zwei von diesen Farben nehmen, sie mischen und so eine neue erzeugen. Man ist immer auf diese 4096 Farben festgelegt. Mehr dazu aber in einem folgenden Abschnitt.

12.1.3 Pointilismus

Eine ähnliche Einschränkung wie bei der Anzahl möglicher Farben gibt es hinsichtlich der Anzahl von einzelnen Punkten, aus denen ein Bild aufgebaut wird. Anders als ein Bild, das man mit Farbe auf einem Blatt Papier gemalt hat, besteht ein Computerbild nämlich nicht aus mit Farbe gefüllten Flächen, sondern aus einer Unzahl von (eventuell verschiedenfarbigen) einzelnen Punkten, die erst dann zu zusammenhängenden Flächen verschwimmen, wenn man etwas vom Bildschirm zurücktritt. Es gibt eine Kunstrichtung, den »Pointilismus«, bei der Bilder ähnlich entstehen. Auch dort werden kleine Punkte in verschiedenen Farben auf die Leinwand plaziert. Mit etwas Abstand werden aus diesen Punktwolken dann farbige Flächen.

Im Gegensatz zum Pointilismus können die Punkte einer Computergrafik aber auch nicht an beliebigen Stellen auf dem Bildschirm liegen, sondern sind in einem ganz bestimmten gitterähnlichen Muster aus Zeilen und Spalten angeordnet. Sie liegen praktisch in einer Art »Raster«, weshalb diese (die heute übliche) Art von Computergrafik auch »Rastergrafik« heißt.

Bei der echten »hochauflösenden« Rastergrafik kann jeder beliebige Punkt des Rasters einzeln mit Farbe versehen werden. Jeder Punkt kann eine andere Farbe als seine »Nachbarn« haben. Zwischen diesen Punkten aber kennt ein Computer nichts. Eine Fläche, die kleiner ist als ein solcher Punkt, kann nicht dargestellt werden.

Dies bedeutet prinzipiell noch keinen Unterschied gegenüber anderen Medien der Grafik. Auch Filmmaterial besitzt eine Körnung, und deckende Farben bestehen aus einzelnen Pigmenten in einer flüchtigen (verdunstenden) Lösung. Im Buchdruck wird auch mit einem Raster gearbeitet, in dem mit vier Grundfarben und kleinen Punkten nahezu beliebige Bilder erzeugt werden. Der Hauptunterschied zwischen diesen Grafikmedien und dem Computerbildschirm liegt nur in der extrem regelmäßigen Anordnung der Punkte und vor allem in ihrer Größe! Während man die Körnung einer Foto-Emulsion nur mit einer starken Lupe erkennen kann, sind die Punkte einer Computergrafik, man nennt sie auch »Pixel«, meist deutlich zu erkennen. Gehen Sie einfach einmal etwas näher an den Bildschirm des Amiga heran und Sie werden sehen, daß die Bilder, die er Ihnen zeigt, aus kleinen, fast rechteckigen Punkten bestehen, die deutlich zu unterscheiden sind.

12.1.4 Das Prinzip der »bitmapped« Grafik

Welche Farbe ein bestimmter Punkt auf dem Bildschirm hat, merkt sich der Computer in seinem Arbeitsspeicher. Wie bei Computern üblich, kodiert er die Farbe als Zahl. Das heißt, für jeden Punkt auf dem Bildschirm reserviert ein Computer, der über Rastergrafik verfügt, einen Platz in seinem Arbeitsspeicher, in dem er jederzeit nachsehen kann, welche Farbe dieser Punkt hat. Um einem Punkt eine andere Farbe zu geben, braucht ein Programmierer nur die diesem Punkt entsprechende Stelle im Speicher zu finden und dort eine andere Zahl einzutragen.

Da Computer sehr sparsam mit ihrem Arbeitsspeicher umgehen müssen, reservieren sie natürlich so wenig Platz wie möglich im Speicher für die Zahlen, die die Farben der einzelnen Bildschirmpunkte sind. Am »sparsamsten« in dieser Hinsicht ist reine Schwarzweiß-Grafik (oder besser zweifarbige Grafik). Hierbei kann jeder Punkt am Bildschirm (jedes Pixel) nur zwei Zustände haben, Schwarz und Weiß (oder Schwarz und Grün, Schwarz und Gelb und so weiter), und der Computer braucht sich nur zwei verschiedene Zahlen für diesen einen Punkt zu merken. Und das wiederum kann er schon mit einem einzigen »Bit«, die kleinsten Speichereinheit, die ein Computer kennt.

Das legendäre Bit – Sie haben bestimmt schon einmal davon gehört – kann genau zwei Zustände annehmen, die man wahlweise mit »An« und »Aus«, »Wahr« und »Falsch«, 1 und 0 oder eben auch mit »Schwarz« und »Weiß« bezeichnen kann. Von diesen Bits, in denen sich der Computer die Farbe eines Pixels merkt, kommt der Name »bitmapped Grafik« (zu deutsch etwa »Bitkarten-Grafik«) für die spezielle Unterart der Rastergrafik, die die meisten grafikfähigen Computer heutzutage benutzen. Der Computer reserviert bei dieser Art Grafik einen bestimmten Bereich in seinem Arbeitsspeicher, der praktisch als Karte für das Bild dient. Genau wie bei einer Karte ein kleiner Punkt einer Ortschaft entspricht, entspricht bei einer Bitmap (Bitkarte) ein Bit einem Pixel am Bildschirm.

0	1	0	1	0	0	0	0	1	0	0	0	0	1	0
1	0	0	1	1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0	1	1	0	0	1
0	0	0	1	1	0	0	0	1	1	0	1	0	1	1
0	1	0	1	0	0	0	0	1	0	0	0	0	1	0
1	0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	0	0	1	1	0	0	0	1	1	0	1	0	1	1
0	1	0	1	0	0	0	0	1	0	0	0	0	1	0
1	0	1	0	1	0	1	0	0	0	1	1	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	0	0	1	1	0	0	0	1	1	0	1	0	1	1
1	0	1	0	1	0	1	0	0	0	1	1	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1	0	0	0

Bild 12.1: *Eine Bitkarte für den Bildschirm*

In Wirklichkeit liegen die Bits für ein Bild im Speicher natürlich nicht in einer rechteckigen Anordnung, sondern hintereinander. Die rechteckige Darstellung hilft einem aber, etwas leichter zu verstehen, welches Bit welchem Bildschirmpunkt entspricht. Wie die verschiedenen Bits hintereinander im Speicher liegen, zeigt die folgende Abbildung. Sie zeigt zugleich, in welcher Reihenfolge der Elektronenstrahl Ihres Monitors oder Fernsehers über den Bildschirm streicht, um so ein Bild zu erzeugen. Wenn Sie den durchgezogenen Linien folgen, durchlaufen Sie die Bits (Kästchen) in genau der Reihenfolge, in der sie hintereinander im Speicher liegen. In genau dieser Reihenfolge liest die Elektronik des Amiga 500 diese Bits und steuert anhand des jeweiligen Inhalts den Elektronenstrahl. Ist das Bit an (=1), wird der Elektronenstrahl verstärkt und der Punkt am Bildschirm hell. Ist das Bit aus (=0), wird der Elektronenstrahl gedrosselt und der Punkt am Bildschirm bleibt dunkel. Die gestrichelten Linien in der Abbildung entsprechen den Zeiten, in denen der Elektronenstrahl immer gedrosselt bleibt, während es zum Anfang einer neuen Zeile geht oder zum Anfang des Bildes, nachdem er es einmal komplett auf den Bildschirm gezeichnet hat.

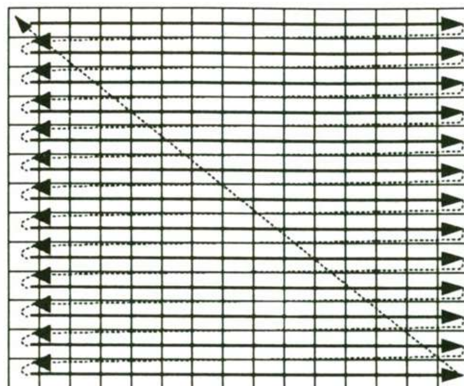


Bild 12.2: Reihenfolge, in der die Bits im Speicher liegen

12.1.5 Farbe ins Bild

Soll ein Computer aber nicht nur zwei Farben am Bildschirm darstellen können, sondern mehr (wie Ihr Amiga 500), muß für jeden Bildschirmpunkt eine größere Zahl gespeichert werden. Auch dabei geht man natürlich so ökonomisch wie möglich vor und teilt einem Bildschirmpunkt immer eine bestimmte Anzahl von Bits zu. Das Bitkarten-Prinzip stimmt also noch, nur daß nun mehrere Bits einem Pixel entsprechen.

Nun müssen Sie ein wenig in die Tiefen der »Dualzahlen« eindringen, der Zahlen, die nur aus Bits oder Nullen und Einsen bestehen. Nur dann können Sie verstehen, wieviele verschiedene Zahlen (Farben) man mit einer bestimmten Anzahl Bits darstellen kann. Sie haben ja bereits gesehen, daß man in einem Bit zwei verschiedene Zustände beziehungsweise zwei Zahlen speichern kann. Man nennt diese beiden Zahlen meist nicht 1 und 2, sondern 0 und 1. Hat man statt einem Bit zwei zur Verfügung, kann man damit schon vier ($2*2$ oder 2^2) verschiedene Zustände speichern, 00, 01, 10 und 11. Dies entspricht den Zahlen 0, 1, 2 und 3. Nimmt man gar drei Bit, reicht dies schon zum Speichern von 8 verschiedenen Zahlen ($2*2*2$ oder 2^3), vier Bit reichen für 16 Zahlen ($2*2*2*2$ oder 2^4) und so weiter.

Diese Art der Kodierung von Zahlen beziehungsweise Farben ist auch der Grund, warum grafikfähige Computer nie 5, 10 oder 100 Farben bieten, sondern immer 2, 4, 8, 16, 32 und so weiter. Der Speicher des Computers wird auf diese Weise einfach am besten ausgenutzt. Mit vier Bit kann man nun einmal 16 Farben darstellen und mit dreien nur 8 Farben. Es gibt keine halben Bits und somit auch keine Möglichkeit, einen Speicherbereich zu reservieren, der genau

10 verschiedene Zahlen speichern kann. Das ist genauso unmöglich wie eine Digitaluhr, deren Sekundenanzeige halbe Sekunden anzeigt.

12.1.6 Wie Farben auf den Bildschirm kommen

Welche Farben kann ein Computer aber überhaupt auf den Bildschirm bringen und wie? Diese Frage ist relativ leicht zu beantworten: Grundsätzlich verwendet ein Computer (-Monitor) nämlich dasselbe Prinzip, um Farben zu zeigen, wie auch Ihr Farbfernseher. Er mischt durch »additive Farbmischung« aus den drei Grundfarben Rot, Grün und Blau die gewünschte Farbe zusammen. Betrachten Sie sich Ihren Monitor oder Farbfernseher einmal näher, dann werden Sie sofort verstehen, wie das gemeint ist. In einer scheinbar weißen Fläche werden Sie dann nämlich drei verschiedenfarbige Arten von leuchtenden Punkten erkennen, rote, grüne und blaue. Bei manchen Bildschirmen sind sie rund, bei anderen schlitzförmig. Wenn alle drei Punkte gleichzeitig leuchten, ergibt das eine weiße Fläche, sobald Sie etwas zurücktreten.

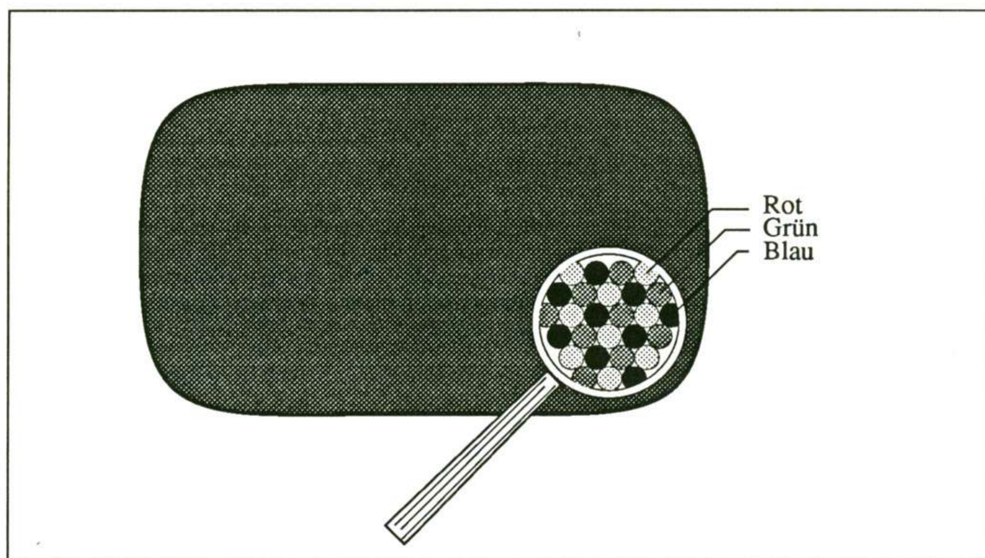


Bild 12.3: Das Grundprinzip des Farbmonitors (und des Farbfernsehens)

Jeder Bildschirmpunkt besteht deshalb aus (mindestens) drei dieser leuchtenden Punkte. Jeder dieser drei Punkte kann von drei verschiedenen Elektronenstrahlen, mit denen ein Farbfernseher arbeitet, getrennt zum Leuchten gebracht werden. Je nachdem, wie hell jeder der drei Leuchtpunkte leuchtet, aus denen sich ein Bildschirmpunkt zusammensetzt, ergeben sich verschiedene Farben. Leuchten alle gleich hell, ergibt das Weiß oder einen Grauton. Sind alle drei Punkte dunkel, bleibt der Punkt schwarz. Durch feine Abstufungen der Leuchtintensitäten kann man alle Farbtöne erzeugen, die man sich nur vorstellen kann. Mit dem Werkzeug

Preferences und den drei Schiebereglern im Hauptfenster können Sie das ausprobieren. Diese drei Schieberegler bestimmen die Rot-, Grün- und Blau-Anteile der vier Workbench-Farben.

12.1.7 Malen nach Zahlen

Im Gegensatz zu einem Farbfernseher kann ein Computer aber nicht beliebige Farben erzeugen. Er arbeitet ja »digital« und speichert zu jedem Bildschirmpunkt eine Zahl, die dessen Farbe bestimmt. Welche der Zahlen oder Bitkombinationen, die für einen Bildschirmpunkt abgespeichert werden, welchen Farben am Bildschirm entsprechen, bleibt weitgehend den Ingenieuren überlassen, die den Computer konstruieren. Vielleicht haben Sie schon einmal die Bilder, mit denen angeblich jeder (vor allem Kinder und Jugendliche) ein großer Maler werden kann, gesehen. Bei diesen Bildern erhält man eine Kiste mit Farben und eine Leinwand oder ein Blatt Papier, auf dem die Konturen des Bildes – ähnlich wie bei einem Bilderbuch – schon vorgegeben sind. Jede Fläche des Bildes trägt eine Zahl und man braucht nun noch den zu dieser Zahl gehörenden Farbtopf aus der Kiste zu nehmen und damit die Fläche auszumalen. »Malen nach Zahlen« heißt das Ganze und obwohl die Bilder, die daraus hervorgehen, meist nicht gerade nach großen Kunstwerken aussehen – das Prinzip funktioniert ganz gut.

So ähnlich verfährt auch der Computer bei mehrfarbiger bitmapped Grafik. Und genau wie beim Malen nach Zahlen muß es nicht unbedingt eine logische Beziehung zwischen bestimmten Zahlen und bestimmten Farben geben. Es reicht, wenn diese Beziehung eindeutig ist und nicht dieselbe Zahl zwei verschiedenen Farben zugeordnet wird. Es gibt eine ganze Reihe von verschiedenen Prinzipien, diese Beziehung logischer aufzubauen.

12.1.8 Achtfarbige Grafik

Bei vielen Computern wird heute ein Prinzip angewendet, bei dem die Farbe jedes Bildpunktes in drei Bits abgelegt wird. Unterschiede gibt es nur dabei, wie die acht Zahlen (bei 3 Bit) als Farben interpretiert werden. Man kann sich natürlich acht beliebige Farben aussuchen und sie fest diesen acht Zahlen zuordnen. Bei einigen Modellen macht man sich auch die Tatsache zunutze, daß Farbfernsehbilder aus den drei Grundfarben Rot, Grün und Blau durch additive Farbmischung zusammengesetzt werden. Jedes der drei Bit, die auf einen Bildschirmpunkt abgebildet werden, wird deshalb einer dieser Grundfarben zugeordnet. Das erste Rot, das zweite Grün und das dritte Blau. Jede Farbe, deren Bit gleich 1 ist, wird dann bei dem jeweiligen Punkt »beigemischt«. Ist das Bit gleich 0, bleibt die Farbe weg. Die acht Farben, die auf diesem Wege gemischt werden können, entsprechen dabei den acht möglichen Kombinationen von drei Bit. Schwarz entspricht zum Beispiel der Bitkombination 000, Weiß der Kombination 111, Rot der Kombination 100, Violett der Kombination 101 und so weiter. Verschiedene Intensitäten der drei Grundfarben sind jedoch nicht möglich. Jede ist entweder an (wird beigemischt) oder aus (wird ganz fortgelassen) und deshalb gibt es nur acht Farben, die sich auf diese Weise mischen lassen.

Ziemlich genau dieses Prinzip wird zum Beispiel bei der Standardfarbgrafik (CGA) der IBM PCs verwendet. Es ist einfach zu realisieren, aber auch recht eingeschränkt. Mehr als acht Farben kann man nämlich so nicht darstellen, und auch diese acht Farben liegen eben »fest« und können nicht geändert werden. Beim Amiga 500 wird ein anderes Prinzip zur Erzeugung

von Farben angewendet, die sogenannte »Farbregister-Technik«, über die Sie in einem späteren Abschnitt noch mehr erfahren werden.

12.2 Grafische Fähigkeiten des Amiga

Nach diesen mehr allgemeinen Ausführungen über die Computergrafik folgen nun aber noch ein paar »harte Fakten« über die Grafikfähigkeiten Ihres Amiga 500.

12.2.1 Bildauflösung

»Bildauflösung« nennt man die Anzahl unterscheidbarer Bildschirmpunkte auf einer bestimmten Strecke. Im übertragenen Sinne nennt man oft auch die Anzahl der Zeilen und Spalten, aus denen das Raster eines grafikfähigen Computers besteht, »Bildauflösung«. Sie ist ein Maß für die »Feinheit« der Grafik, zu der ein Computer fähig ist und wird getrennt in vertikale Auflösung (=Anzahl unterscheidbarer Zeilen in einem Bild) und horizontale Auflösung (=Anzahl unterscheidbare Punkte pro Zeile).

Der Amiga bietet die Wahl zwischen zwei vertikalen Auflösungen, 256 oder 512 Zeilen, und zwei horizontalen Auflösungen 320 oder 640 Punkten. Kombiniert ergibt dies vier verschiedene Bildauflösungen mit mindestens 81 920 und höchstens 327 680 Punkten auf dem Bildschirm. Das hört sich nach viel an, ist es aber nicht. Selbst das relativ körnige Bild einer Pocketkamera hat mehr Punkte. Die folgende Tabelle gibt einen Überblick über die verschiedenen Auflösungen, und die jeweilige Anzahl von Punkten.

Horizontale Auflösung	Vertikale Auflösung	Anzahl Punkte
320	256	81 920
640	256	163 840
320	512	163 840
640	512	327 680

Tabelle 12.1: Grafikauflösungen des Amiga 500

Die in den USA verkauften Modelle des Amiga, die an den PAL-Fernsehstandard angepaßt sind, besitzen nur eine vertikale Auflösung von 200 beziehungsweise 400 Zeilen. Viele der derzeit erhältlichen Programme (die ja fast alle aus den USA kommen) unterstützen die höhere vertikale Auflösung der europäischen Modelle aber nicht. Ich werde im folgenden aber trotzdem von 256 oder 512 Zeilen vertikaler Bildschirmauflösung ausgehen.

Aus dieser Tabelle ergeben sich aber noch andere Konsequenzen. Die verschiedenen Mengen von Punkten, die der Amiga 500 in den einzelnen Betriebsarten seiner Grafik separat färben

kann, nehmen ja immer denselben Platz auf dem Bildschirm ein. So ist eine Grafik in einem Modus mit 256 Zeilen nicht etwa halb so hoch wie eine Grafik in einem Modus mit 512 Zeilen. Statt dessen ist die zweite Grafik »feiner«, weil auf derselben Strecke, auf der vorher 20 Punkte lagen, im 512er-Modus nun 40 Punkte liegen, mit denen sich feinere Details darstellen lassen. Daraus ergibt sich aber auch zwingend, daß die Pixel (Bildschirmpunkte) nicht in allen Grafik-Betriebsarten gleich groß sind. Die folgende Gegenüberstellung zeigt Ihnen (in stark vergrößerter Form) die vier verschiedenen Pixelgrößen, mit denen Sie auf dem Amiga 500 arbeiten können.

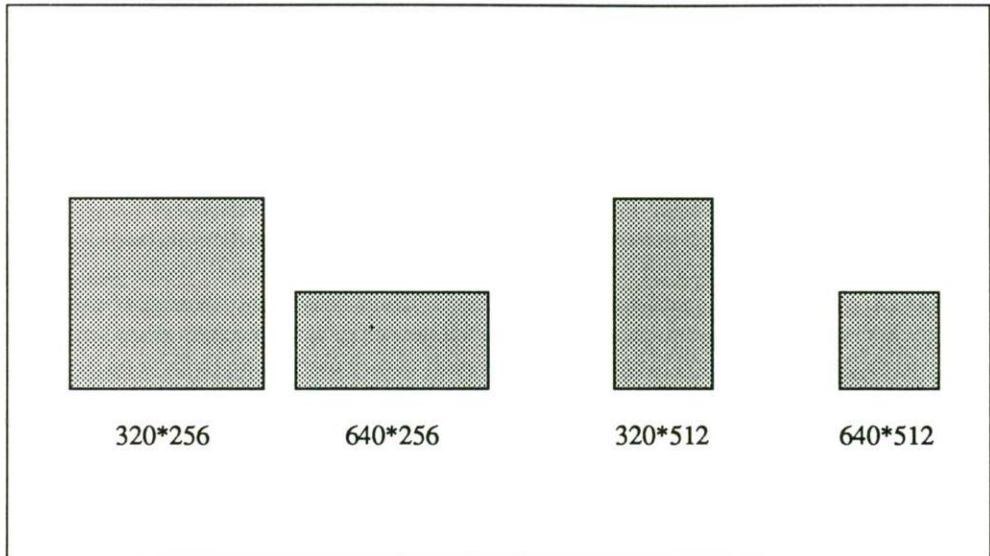


***Bild 12.4:** Die vier verschiedenen Pixelgrößen des Amiga 500*

Die hier angedeutete Rechteckform der Pixel ist natürlich übertrieben. In Wirklichkeit sind die Ecken der Rechtecke so stark abgerundet, daß es auf manchen billigen Monitoren fast Ovale werden. Beachten Sie aber bitte auch, daß die Pixel nicht immer quadratisch sind. Wenn Sie ein Rechteck auf den Bildschirm zeichnen, das 100 Pixel breit und hoch ist, so wird es in zwei Grafik-Betriebsarten annähernd quadratisch (in einem jedoch nur halb so groß) sein und in den beiden anderen Betriebsarten rechteckig. Das wird dann wichtig, wenn Sie selbst Grafikprogramme schreiben oder mit einem Malprogramm ein Bild in einer anderen Auflösung betrachten als die, in der es gezeichnet wurde.

12.2.2 Die Farbpalette

Bei der niedrigsten Bildauflösung kann der Amiga bis zu 32 verschiedene Farben gleichzeitig auf dem Bildschirm darstellen, bei der höchsten Auflösung mit 512 Zeilen zu 640 Punkten sind es immerhin noch 16. Das Programm Deluxe Paint benutzt standardmäßig zum Beispiel eine

Bildschirmauflösung von 200*320 Punkten und 32 Farben. Die Workbench hingegen arbeitet mit einer Bildschirmauflösung von 256*640 Punkten und nur vier Farben.

Wie vereinbart sich die eben genannte Zahl von maximal 32 möglichen Farben nun aber mit der weiter oben genannten Zahl von 4096 Farben, die der Amiga darstellen kann? Eine der beiden Zahlen scheint ja wohl (auf den ersten Blick) falsch zu sein.

Dies ist jedoch nicht der Fall. Die beiden Aussagen beziehen sich auf zwei unterschiedliche Tatbestände, auf die ich nun etwas näher eingehen werde: Der Amiga ist fähig, 4096 verschiedene Farben darzustellen. Von diesen kann er aber immer nur maximal 32 gleichzeitig am Bildschirm zeigen. Man muß also aus den 4096 Farben eine Palette mit bis zu 32 Farbklecksen zusammenstellen und mit diesen ausgewählten Farben das gezeigte Bild malen. Welche 32 Farben Sie hierzu benutzen wollen steht Ihnen völlig frei – nur mehr als 32 Farben dürfen es nicht sein.

Der Grund für diese Einschränkung ist relativ einfach. Wie ja weiter oben bereits erläutert wurde, merkt sich der Amiga für jeden Bildpunkt eine Zahl in seinem Arbeitsspeicher, die bestimmt, welche Farbe dieser Punkt trägt. Die Zahl ist dabei praktisch ein Kode für die gewünschte Farbe. Um 4096 verschiedene Farben kodieren zu können, benötigt ein Computer 12 Bit ($2^{12} = 4096$). Der Amiga würde also bereits bei der niedrigsten Bildschirmauflösung 81 920*12 = 983 040 Bit benötigen, nur um den Bildschirminhalt abzuspeichern. Dies entspricht über 120 Kbytes und damit fast einem Viertel des Arbeitsspeichers eines Amiga 500 ohne Speichererweiterung. Übrigens sprechen auch andere technische Gründe neben dieser Speicherplatzverschwendung gegen so viele Farbmöglichkeiten.

Um 32 Farben darzustellen, benötigt man hingegen nur 5 Bit; nicht einmal halb so viele wie für die volle Farbenzahl. Um mit nur 5 Bit, also einer Zahl zwischen 0 und 31, trotzdem eine von 4096 Farben bestimmen zu können, verwendet der Amiga eine zusätzliche Stufe der Kodierung. Die Zahl, die er sich zu einem bestimmten Bildschirmpunkt merkt, bestimmt nicht direkt die Farbe, die dieser Punkt hat, sondern ist nur die Nummer eines Farbtopfs in der aktuellen Palette. Die Palette wiederum enthält 32 der 4096 möglichen Farben (diese Farben müssen dazu natürlich als 12 Bit abgespeichert werden). Der Computerfachmann nennt die Farbtöpfe in der Palette übrigens auch »Farbregister«.

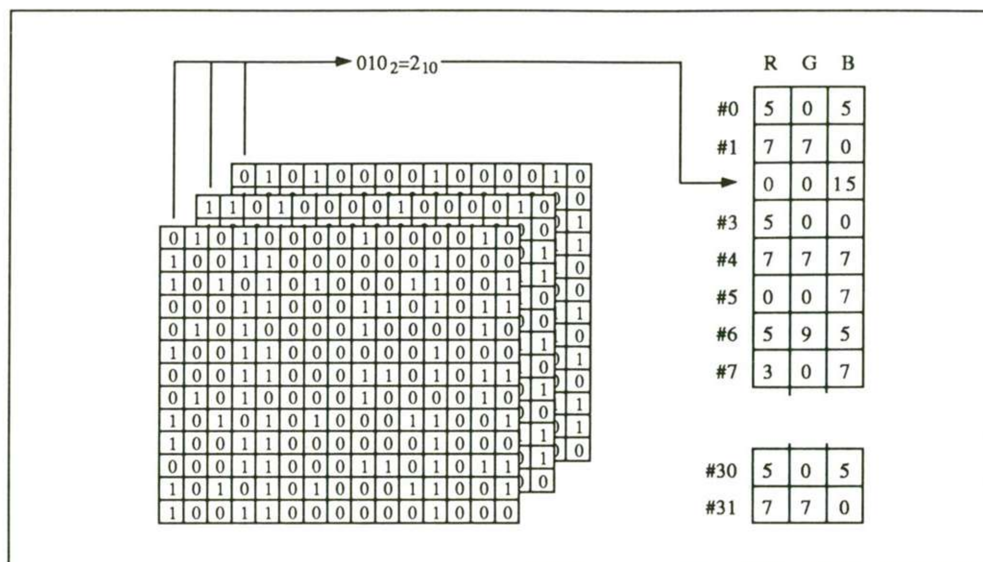


Bild 12.5: *Bestimmung der Farbe eines Bildschirmpunktes*

Wie an dem obigen Bild zu sehen ist, liegen die Bits, die für einen Punkt »zuständig« sind, nicht direkt hintereinander. Die drei Bit, die hier zur Auswahl einer Farbe in der Farbpalette dienen, liegen in drei verschiedenen Speicherbereichen, die hier hintereinanderliegend dargestellt wurden. Zu diesen zusammenhängenden Speicherbereichen, in denen jeweils die ersten, zweiten und dritten Bits aller Pixel liegen, sagt man auch »Bitebenen« oder »Farbebenen«. Wenn drei solche Ebenen für eine bestimmte Grafik benötigt werden, sagt man oft auch die Grafik oder der Bildschirm ist »drei Bit« oder »drei Bitebenen tief«. Aus der Tiefe einer Grafik und ihrer Auflösung ergibt sich der Speicherbedarf eines Bildes. Dazu müssen Sie die Anzahl der Punkte in einer Grafik mit der Tiefe (der Anzahl Bits pro Punkt) multiplizieren und dann durch acht teilen (acht Bit = ein Byte). Die folgende Tabelle nennt einige Beispiele für den Speicherbedarf (in Bytes) von Bildern bei verschiedenen Auflösungen. (Zu Ihrer Information ist darin auch noch einmal die maximal mögliche Zahl unterschiedlicher Farben bei der jeweiligen Tiefe angegeben.)

Auflösung	Tiefe	Anzahl Farben	Speicherbedarf
320 * 256	1	2	10 240
640 * 256	2	4	40 960
320 * 256	3	8	30 720
640 * 512	4	16	163 840
320 * 512	5	32	102 400

Tabelle 12.2: *Grafikauflösungen und ihr Speicherbedarf*

Damit aber genug der Wortklauberei. Der eben beschriebene »Umweg« über die Farbpalette hat nämlich eine interessante Konsequenz, die im folgenden Abschnitt beschrieben wird. Damit sind Effekte ermöglicht, die mit echter Farbe und Papier auf keinerlei Weise zu erreichen sind.

12.2.3 Nachträgliches Ändern der Palette

Um die Farbe eines Pixels zu ändern, gibt es auf dem Amiga nämlich nicht nur eine Möglichkeit, sondern gleich zwei. Zunächst einmal kann man natürlich die Zahl (Bitkombination) im Speicher ändern, die die Farbe des Pixels bestimmt. Dieses Pixel erscheint dadurch in einer anderen der 32 Paletten-Farben am Bildschirm. Dieser Vorgang entspricht praktisch dem Ändern der Farbe eines Punktes auf dem Papier durch erneutes Überstreichen mit einer anderen Farbe.

Alternativ dazu kann man aber auch die Farbe in der Palette ändern, die durch die Zahl ausgewählt wird, die im Computer für dieses Pixel gespeichert wird (siehe oben). Diese Art der Änderung der Pixelfarbe hat jedoch einen Nachteil (oder Vorteil – wie man es sieht). Im selben Augenblick, in dem an einer Stelle in der Palette eine neue Zahl (= Farbe) eingetragen wird, ändern alle Bildschirmpunkte, die ihre Farbe aus diesem Topf der Palette bezogen haben, ihre Farbe! Wenn Sie also die Farbe 1 in der Palette von Blau nach Braun ändern, werden alle Punkte des Bildes, die mit dem Blau der Farbe 1 gefärbt waren, schlagartig braun.

Dies liegt daran, daß ein Computer das Bild, das er am Bildschirm zeigt, nicht einmal zeichnet und dann »stehenläßt«. Wie es dem Prinzip des Monitors (und auch Ihres Fernsehers) entspricht, muß das Bild etwa 50mal in der Sekunde neu gezeichnet werden. Ändert man eine Farbe in der Palette und der Amiga sieht beim nächsten Zeichnen des Bildes in der Palette nach, welche Farbe der Zahl entspricht, die er für ein Pixel abgespeichert hat, ist dies die neue Farbe – folglich kommt es dann am Bildschirm zu einer Farbänderung, obwohl an dem Pixel nichts geändert wurde.

Stellen Sie sich die Möglichkeiten vor, die diese Methode bietet! Sie können eine Grafik zum Beispiel zunächst in nahezu beliebigen Farben zeichnen und sich hinterher noch einmal in Ruhe vor den Bildschirm setzen und nun die Palette bearbeiten. Sie können blitzschnell neue

Farbkombinationen austesten, auf Vorschläge in dieser Hinsicht von Freunden oder Kunden eingehen und so weiter.

Eine weitere, sehr interessante Anwendung dieses Konzepts ist die »Farbpalettenanimation« (engl. color register animation) oder »Farbenrotation«. Mit diesem Kniff kann man die Illusion von Bewegung erzeugen, indem man den Inhalt der Farbpalette ändert und so recht rasch auch die Färbung großer Flächen auf dem Bildschirm auswechseln kann.

Die Idee der Farbpalettenanimation ist relativ simpel: Ein kleines Programmstück sorgt dafür, daß eine Gruppe von Farben in der Palette automatisch in einem bestimmten Rhythmus geändert wird. Diese Gruppe tauscht die Farben zyklisch aus. Jeweils eine Farbe wird zum nächsten Farbtopf weitergereicht, und dieser gibt vorher seine Farbe an den übernächsten weiter. Der letzte Farbtopf dieser Gruppe schließlich gibt seine Farbe an den Ersten der Gruppe weiter.

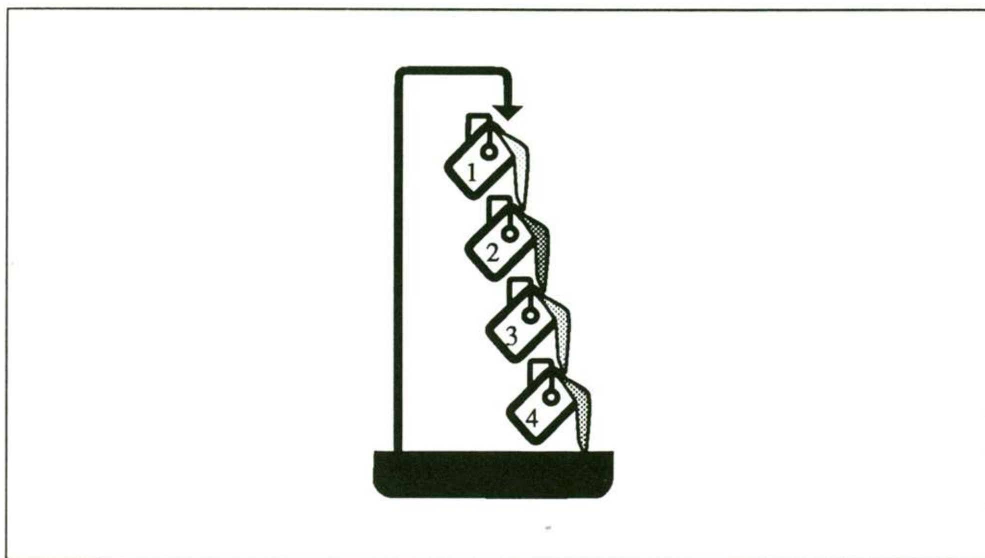


Bild 12.6: *Das Prinzip der Farbpaletten-Animation*

Enthalten die Farbtöpfe, die ihre Farben austauschen, zum Beispiel verschiedene Abstufungen einer Farbe, so wirkt der Farbwechsel unter Umständen wie eine Bewegung der mit den dunklen Farben bemalten Flächen. Mit ein wenig Geschick können Sie auf diese Weise die tollsten Effekte erzielen. Ein berühmtes Beispiel hierfür ist das Bild eines Wasserfalls, bei dem das Wasser in verschiedenen Blautönen gemalt wurde. Läßt man diese Blautöne nun »rotieren«, so scheint der Wasserfall in dem vorher ruhenden Bild auf einmal zu fließen.

Alle Malprogramme für den Amiga 500 bieten Möglichkeiten zur Farbpalettenanimation. Im Programm GraphiCraft startet man die Farbpaletten-Animation zum Beispiel mit dem Befehl *Cycle Colors* (engl. für Farben rotieren) im Menü *Special*. Die Animation kann wieder gestoppt werden, indem der Befehl *Cycle Colors Off* aufgerufen wird. In Deluxe Paint (I und II) genügt ein Druck auf die [Tab]-Taste, um die Bewegung der Farben in der Palette in Gang zu setzen oder zu stoppen. Bei beiden Programmen werden eine ganze Reihe von Beispielbildern mitgeliefert, anhand deren man ausprobieren kann, welche vielfältigen Möglichkeiten ein so simples Prinzip wie die Farbpalettenanimation schon bietet.

12.2.4 Der Interlace-Modus

Computergrafik hat aber nicht nur ihre schönen und interessanten Seiten. Manche Eigenschaften der Amiga-Grafik, die sich aus technischen Zwängen ergeben, die man nicht abstellen kann, sind geradezu unangenehm. Dazu gehört auch eine Eigenheit der hohen vertikalen Auflösung von 512 Punkten, die Sie bestimmt gleich bemerkt haben, wenn Sie die Workbench einmal in dieser Auflösung betrachtet haben: das Bild flimmert dann fast unerträglich. (Sie können die Workbench auf hohe Auflösung stellen, indem Sie *Preferences* aufrufen und den Schalter *Interlace* aktivieren, dann Preferences mit einem Klick auf *Save* verlassen und einen Neustart machen.)

Die vertikale Auflösung beträgt auf dem Bildschirm normalerweise ja 256 Punkte. Dies ist die Grenze, die von einigermaßen preiswerten Monitoren und Farbfernsehern gesetzt wird und bei der noch ein relativ flimmerfreies Bild möglich ist. Die Bildwiederholfrequenz beträgt bei dieser Auflösung 50 Hz. Das heißt, 50mal pro Sekunde wird das Bild neu gezeichnet. Das ist schneller als das menschliche Auge es bemerken kann, weshalb von diesem andauernden Neuzeichnen normalerweise nichts zu merken ist. Im *Interlace*-Modus müssen jedoch 512 (oder bei amerikanischer Software 400) Zeilen auf den Bildschirm geschrieben werden. Und das schafft der Amiga nicht in derselben Zeit wie das Zeichnen von 256 Zeilen.

Im Interlace-Modus wird das 512 Zeilen hohe Bild deshalb in zwei »Halbbilder« mit jeweils 256 Zeilen aufgeteilt. Diese Halbbilder werden mit einer Bildwiederholfrequenz von 50 Hz abwechselnd auf den Bildschirm geschrieben. Dabei ist jedes Halbbild aber nur jedes zweite Mal »dran«. Für ein Halbbild gilt also eine Bildwiederholfrequenz von nur 25 Hz. Und das kann das menschliche Auge (leider) noch wahrnehmen – es flimmert. Damit Sie mich aber nicht falsch verstehen: die beiden Halbbilder bestehen natürlich nicht aus der oberen und unteren Bildhälfte. Ein Halbbild enthält statt dessen jede zweite Zeile des Bildes. Das erste Halbbild besteht also aus allen ungeraden, das andere aus allen geraden Zeilen des Vollbilds mit 512 Zeilen. Die folgende Skizze zeigt dieses Prinzip in schematischer Form (dabei werden allerdings weit weniger als 512 Zeilen verwendet).

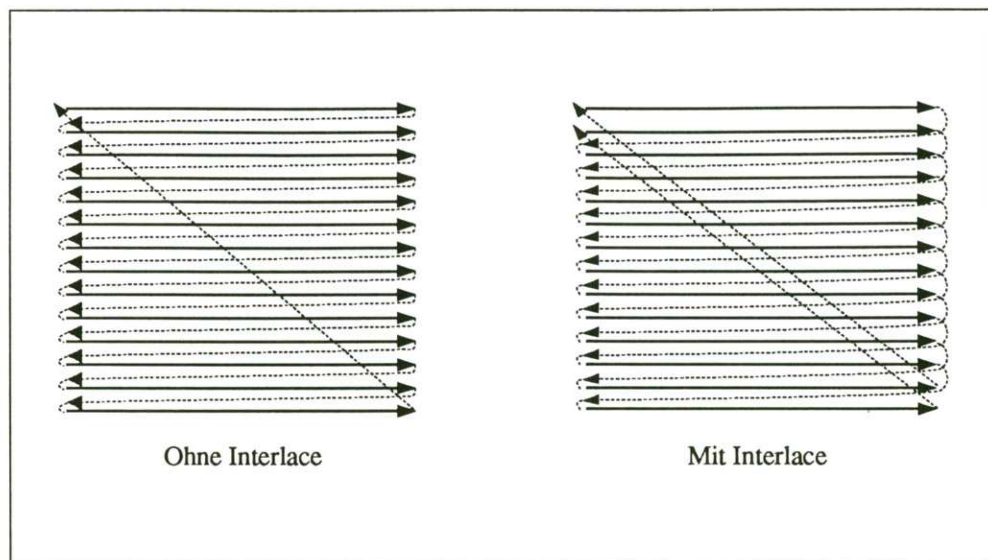


Bild 12.7: Wie der Elektronenstrahl des Monitors ein Interlace-Bild zeichnet

Der Interlace-Modus ist übrigens etwas ganz Alltägliches. Auch das normale Fernsehen basiert auf diesem Prinzip. Das Flimmern wirkt beim Fernsehen nur deshalb nicht so schlimm, weil die Kontraste nicht so hart sind wie bei einem computererzeugten Bild, weil das Bild sich sowieso schneller ändert und das Flimmern deshalb nicht so auffällt und schließlich auch deshalb, weil Sie nicht so nahe vor einem Fernseher sitzen wie vor dem Monitor des Amiga.

Der Interlace-Modus hat aber trotz dieser unangenehmen Eigenarten durchaus seine Daseinsberechtigung. Wenn Sie nämlich unbedingt möglichst viele Informationen auf dem Bildschirm sehen wollen, bietet er zum Beispiel die doppelte Anzahl von Zeilen. Und wenn Sie Fotos vom Bildschirm machen wollen, spielt das Flimmern sowieso keine Rolle, da das Foto ja hinterher nicht mehr flimmert. Ein anderer Weg, das Flimmern zu umgehen ist es, einen Monitor zu verwenden, dessen Phosphorschicht länger nachleuchtet. Monochrome Monitore dieser Art gibt es bereits in reichlicher Auswahl. Bei Farbmonitoren ist diese Art Phosphor noch selten. Das Flimmern des Interlace wird damit auf ein Minimum reduziert. Leider aber ziehen schnell bewegte Objekte (wie der Mauszeiger) auf solchen Monitoren dann auch deutliche Spuren – wie Kometenschweife – hinter sich her, die auch sehr störend wirken können.

12.3 Ungewöhnliche grafische Fähigkeiten des Amiga 500

Die bislang beschriebenen Aspekte der Amiga-Grafik waren mehr oder weniger »normal« oder »üblich«. Viele andere Computer haben in grafischer Hinsicht solche oder ähnliche

Eigenschaften. Der Amiga 500 hat allerdings in mancher Hinsicht bessere »Eckdaten«. So ist es zum Beispiel bei anderen preiswerten Computern recht unüblich, 32 Farben aus einer Palette von 4096 gleichzeitig am Bildschirm verwenden zu können. In den folgenden Abschnitten werden aber auch einige Aspekte der Amiga-Grafik behandelt, die recht »ungewöhnlich« sind. Einige der beschriebenen Fähigkeiten bietet überhaupt kein anderer Computer.

12.3.1 HAM: Mehr Farben aus dem Computer

Die Grafik des Amiga benötigt – wie oben gesehen – maximal 5 Bit pro Bildschirmpunkt (Pixel), um 32 verschiedene Farben gleichzeitig am Bildschirm darzustellen. Es gibt aber auch einen Modus, der 6 Bitmaps benötigt und theoretisch alle 4096 möglichen Farben gleichzeitig am Bildschirm zeigt. Dieser Modus heißt »Hold-and-Modify-Modus«, oder kurz »HAM-Modus«. Der Name bedeutet soviel wie »festhalten und modifizieren«, und genau das macht die Video-Hardware in diesem Modus.

Die sechs Bit, die für jeden Bildschirmpunkt in diesem Modus zur Verfügung stehen, werden dabei in einer speziellen Weise genutzt, die vom üblichen Bitmap-Prinzip, das weiter oben beschrieben wurde, etwas abweicht. Die Bits Nummer 5 und 6 bestimmen in diesem Modus bei jedem einzelnen Pixel, wie die restlichen Bits (Nummer 1 bis 4) interpretiert werden.

Ist die Bitkombination der Bits 5 und 6 gleich 00, so werden die restlichen 4 Bit als die Nummer eines Farbregisters interpretiert, das die Farbe dieses Punktes enthält. In diesem Fall wirkt der HAM-Modus wie ein etwas speicherverschwendender Grafikmodus mit 4 Bitebenen (= 16 Farben).

Ist die Bitkombination 6-5 aber gleich 01, 10 oder 11, tritt der eigentliche HAM in Aktion. Im Fall 01 wird erst einmal die Farbe festgehalten (engl. *to hold = halten*), die der unmittelbar vor (links neben) dem aktuellen Punkt liegende Bildpunkt hatte. Die Blau-Intensität dieser Farbe wird jedoch verworfen und durch den Wert ersetzt, der in den Bitmaps 1 bis 4 für den neuen Punkt eingetragen ist. Ist diese Bitkombination zum Beispiel 0000, dann wird dem neuen Punkt überhaupt kein Blau mehr beigemischt. Die Bits Nummer 1 bis 4 modifizieren (engl. *to modify = modifizieren*) also den Blauwert des vorangegangenen Punktes und ergeben damit die neue Farbe für den aktuellen Punkt.

Ist die 6-5-Bitkombination gleich 10, wird auch die Farbe des »Vorgängerpunktes« festgehalten. Dann aber ersetzen die Bits 1 bis 4 des neuen Punktes die Rot-Intensität der Farbe des Vorgängerpunktes.

Und wenn die Kombination gleich 11 ist, werden die Bits 1 bis 4 des neuen Punktes dazu benutzt, die Grün-Intensität des Vorgängerpunktes zu modifizieren.

Auf diese Weise können alle 4096 Farben erzeugt werden, zu denen der Amiga 500 überhaupt fähig ist. Es ist allerdings nicht möglich, jeden beliebigen Punkt auf jede beliebige Farbe zu setzen. Spätestens nach drei Punkten kann man aber jede beliebige Farbe erhalten, indem man nacheinander alle drei Komponenten der Pixelfarbe, Rot, Grün und Blau austauscht. Die 16 Farben, die man über ein Farbregister direkt wählen kann (mit der Bit-Kombination 00 der Bits 6 und 5), können natürlich auch noch (möglichst geschickt) eingesetzt werden. Es gibt

inzwischen sogar schon zwei Malprogramme auf dem Markt, mit denen Sie Bilder in diesem Modus erstellen können (sie heißen »DigiPaint« und »Prism«). Weiterhin gibt es eine Reihe zusätzlicher Peripheriegeräte, mit denen Sie Bilder mit einer Videokamera aufnehmen und mit HAM-Modus auf dem Amiga 500 darstellen können. Die so zustande kommenden Bilder sehen erstaunlich gut aus – wovon Sie sich auf den Farbseiten dieses Buches überzeugen können.

12.3.2 Grafiken, die größer als der Bildschirm sind

Der Amiga 500 bietet aber nicht nur die Möglichkeit, unglaublich viele Farben gleichzeitig in einer Grafik zu verwenden, sondern erlaubt auch die mühelose Bearbeitung von Grafiken, die größer sind als der gesamte Bildschirm. Solche Grafiken werden in riesigen Bitmaps gespeichert, von denen nur ein rechteckiger Ausschnitt (eine Art Fenster) am Bildschirm zu sehen ist.

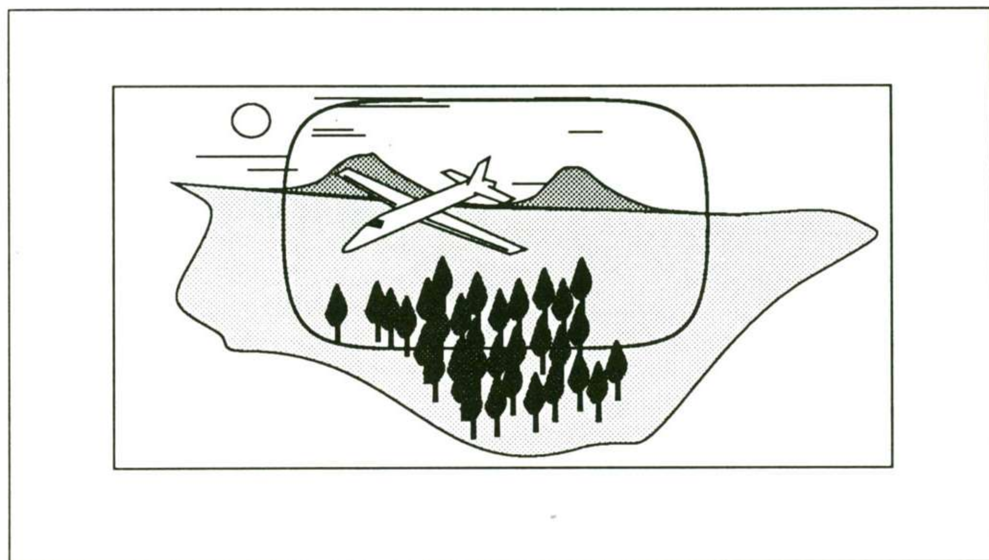


Bild 12.8: Der Bildschirm als Fenster in eine übergroße Grafik

Das »Verschieben« des Bildschirms beziehungsweise des sichtbaren Ausschnitts über diese Riesengrafik ist für Programme sehr einfach. Sie brauchen dazu nur den Inhalt einiger weniger Speicherstellen zu ändern (für Fachleute: diese Speicherstellen sind natürlich Register des Grafikchips *Denise*). Die Verschiebung geht deshalb sehr schnell und ohne das – auf anderen Computern in solchen Fällen übliche – Flimmern und Rucken vonstatten.

12.3.3 Zwei Bilder übereinander: der Dual-Playfield-Modus

Und schließlich verdient (mindestens) noch ein weiterer Aspekt der Amiga-Grafik Erwähnung. Erstens weil er ein ungewöhnliches Konzept darstellt, das bei vergleichbaren Geräten kaum zu finden ist. Und zweitens, weil ihn zumindest ein faszinierendes Programm, Deluxe Video, intensiv nutzt. (Deluxe Video ist ein Programm, mit dem Sie selbst kleine Zeichentrickfilme erstellen können.)

Der Amiga besitzt nämlich einen Betriebsmodus, den sogenannten »Dual-Playfield-Modus«, in dem zwei sich überlagernde Bilder gleichzeitig am Bildschirm dargestellt werden. Der Eindruck, der dabei entsteht, ist ähnlich dem einer transparenten, teilweise bemalten Folie, die über eine andere Zeichnung gelegt wird. An den Stellen, an denen die Folie bemalt ist, wird das dahinterliegende Bild verdeckt. An den Stellen, an denen sie frei (transparent) ist, sieht man das darunterliegende Bild.

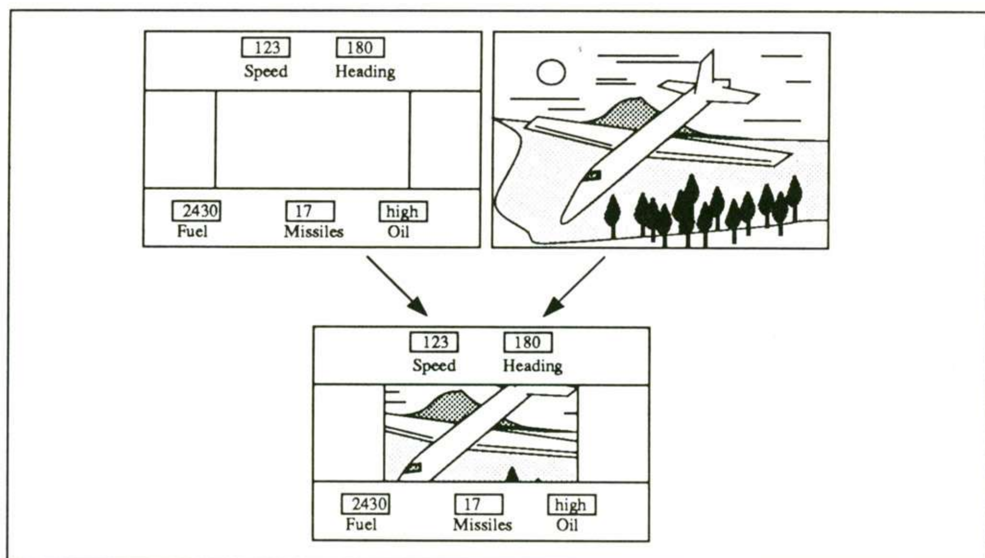


Bild 12.9: *Zwei übereinanderliegende Bilder*

Der Vorteil dieser Fähigkeit des Amiga liegt darin, daß beide Bilder völlig unabhängig voneinander sind. Jedes Bild kann zum Beispiel getrennt verändert und verschoben werden, ohne daß das andere davon beeinträchtigt wird. Beide Bilder haben eine eigene Farbpalette mit bis zu acht verschiedenen Farben und können auch getrennt voneinander ausgetauscht werden.

Man kann sich leicht vorstellen, welche Möglichkeiten das gerade für Trickfilme und Spiele bietet. Das hintere Bild enthält den relativ statischen Hintergrund eines Bildes (der vielleicht

langsam von links nach rechts wandert), das andere die beweglichen Elemente des Vordergrunds, Personen, Tiere, Fahrzeuge, Wolken und so weiter.

12.3.4 Sprites & BOBs

Der Dual-Playfield-Modus bietet schon eine Ahnung von den Eigenschaften, die der Amiga 500 speziell für die bewegte Computergrafik mit sich bringt. Noch »beweglicher« sind aber »Sprites« und »BOBs«. Sprites sind selbständige kleine Bilder, die vom großen Hintergrundbild unabhängig sind. Ähnlich wie das obere Bild im Dual-Playfield-Modus sind sie völlig getrennt vom darunterliegenden Bild.

Ein solches Sprite ist also ein kleines Bild, das maximal 16 Pixel breit und beliebig hoch sein kann (maximal natürlich höchstens nur so hoch wie das Hintergrundbild). Der Mauszeiger ist zum Beispiel ein kleines Sprite.

Um ein Sprite an einer bestimmten Stelle des Bildschirms erscheinen zu lassen, braucht man nur die X- und Y-Position der Stelle, an der die obere linke Ecke des Sprites sitzen soll, in eine bestimmte Speicherstelle (für Fachleute: in ein Register von Denise) schreiben. Ändert man die Werte in diesen Speicherstellen, taucht das Sprite sofort an der neuen Stelle auf. Eine »Bewegung« dieses kleinen Bildes ist so sehr leicht und auch sehr schnell möglich.

Eine gravierende Einschränkung der Sprites ist allerdings ihre Zahl: es gibt nur acht Stück. Diese acht Sprites können jeweils maximal vier verschiedene Farben verwenden. Es gibt auch die Möglichkeit 16 Farben bei jedem Sprite zu verwenden – dabei halbiert sich die Anzahl möglicher Sprites aber auf vier. Acht oder gar nur vier bewegliche Objekte sind für viele Zwecke zu wenig. Deshalb gibt es zwar nur acht beziehungsweise vier Hardware-Sprites, man kann jedes aber mehrmals benutzen. Ein und dasselbe Sprite kann mehrmals in verschiedenen Formen **untereinander,** aber nicht nebeneinander auf dem Bildschirm erscheinen. Wer wissen möchte, woher die Bedingung »untereinander« kommt, muß sich sehr intensiv mit der Amiga-500-Hardware beschäftigen. Für die meisten Anwendungsfälle genügt es, zu wissen, daß es diese Einschränkung gibt und daß ohne Einschränkungen nur acht Sprites verwendet werden können.

Trotz vieler Tricks und Kniffe bleiben Sprites in ihren Möglichkeiten doch recht »beschränkt«. Es gibt nur 8 Stück davon, sie können nur 16 Punkte breit (schmal) werden und verfügen nur über wenige Farben. Alle diese Einschränkungen kennen BOBs nicht. Im Gegensatz zu Sprites sind die BOBs aber kein Konzept der Amiga-Hardware, sondern der Amiga-Software. Diese Software nutzt allerdings intensiv ein besonderes Stück Hardware, den »Blitter«, von dem die BOBs (***Blitter Objects***) auch ihren Namen haben. Über diesen Blitter werden Sie im übernächsten Kapitel noch mehr erfahren.

BOBs verhalten sich nahezu wie Sprites. Es sind kleine Grafiken, die man schnell und einfach verschieben kann, ohne die »darunterliegende« Grafik zu beeinträchtigen. BOBs können im Gegensatz zu Sprites aber genausogroß werden und genausoviele Farben verwenden wie die dahinterliegende Grafik auch. Sie sind im allgemeinen (jedoch nicht zwangsläufig) aber auch recht klein.

Bei BOBs wird eine kleine Bitmap, die das entsprechende Bild enthält, wirklich in die Bitmap des Hintergrundbildes hineinkopiert. Vorher wird aber der alte Inhalt dieser Bitmap in einem Puffer gesichert. Wird das BOB bewegt, so wird diese gesicherte Kopie des alten Inhalts wieder in die Bitmap des Hintergrundbildes zurückgeschrieben und das BOB an der neuen Stelle gezeichnet – und davor natürlich wieder der Inhalt des Hintergrundbildes an der neuen Stelle gesichert. Sie werden sich nun vielleicht fragen, wieso man sich überhaupt mit Sprites abgibt, wo es doch die viel flexibleren BOBs gibt? Die Antwort darauf ist recht einfach: BOBs sind viel langsamer und kosten viel mehr Speicherplatz als Sprites!

Das Bewegen eines BOBs geschieht nämlich durch Software (Programme). Und eine Computer-Grundregel lautet: »Hardware ist schneller (als Software)!« Je größer ein BOB wird, desto größer werden die Unterschiede zu einem Sprite. Wenn sich mehrere große BOBs über den Bildschirm bewegen, so werden diese Bewegungen schon recht langsam und die BOBs fangen an zu flimmern. BOBs sind also einerseits wesentlich flexibler und vielseitiger als Sprites, andererseits aber bei weitem nicht so »beweglich« wie diese.

12.3.5 AnimObjects

BOBs und Sprites eignen sich ideal für die Darstellung bewegter Dinge in einem Programm. Mit beiden Arten von beweglichen Objekten kann man Bilder (mit verschieden großen Einschränkungen) recht mühelos über den Bildschirm schieben. Es reicht jedoch für die Simulation echter Bewegung meist nicht aus, einfach ein »starres Bild« zu verschieben. Wenn in einem Zeichentrickfilm zum Beispiel ein Männlein durch das Bild geht, wird auch nicht ein Bild dieses Männleins unverändert verschoben. An jeder Stelle taucht statt dessen ein anderes Bild des Männleins auf, das verschiedene Phasen der Gehbewegung zeigt. Bein- und Armhaltung ändern sich ständig – wiederholen sich aber natürlich nach einer Anzahl von Bewegungsschritten.

Exakt solche Effekte sind mit AnimObjects möglich. AnimObjects sind eine Reihe von BOBs oder Sprites, die mit einigen zusätzlichen Daten zu einem Objekt zusammengefaßt werden. Ein AnimObject kann dann wie ein Sprite über den Bildschirm bewegt werden, wobei parallel zur Bewegung automatisch ein BOB oder Sprite gegen das nächste ausgetauscht wird.

Die Reihenfolge und Geschwindigkeit, in der die BOBs oder Sprites gegeneinander ausgewechselt werden, ist vom Programmierer frei wählbar. Gleichzeitig kann er dem AnimObject noch eine Geschwindigkeit mitteilen, mit der die BOBs sich über den Bildschirm bewegen sollen und eine Beschleunigung, die angibt, mit welcher Rate sich diese Geschwindigkeit ändern soll.

Ein AnimObject könnte also die verschiedenen Bewegungsphasen einer gehenden Person enthalten. Je mehr Phasen man verwenden würde (und je mehr Speicher man dafür verwenden würde), desto flüssiger würde diese Bewegung wirken. Gibt man dem AnimObject dann eine Geschwindigkeit, so könnte diese Person langsam oder schnell über den Bildschirm schreiten.

12.3.6 Kollisionserkennung

Wenn sich Objekte über den Bildschirm bewegen, ist es oft wichtig, rechtzeitig zu erkennen, wann eines mit dem anderen »zusammenstößt« oder an gewisse Grenzen (zum Beispiel den Rand des Bildschirms oder eines Fensters) stößt. Die Software und die Hardware des Amiga 500 arbeiten deshalb zusammen und überprüfen nach jeder Bewegung eines Objektes (ob BOB, Sprite oder AnimObject spielt dabei keine Rolle), ob eine solche Kollision aufgetreten ist. Dabei kann der Programmierer bestimmen, welche Kollisionen für ihn von Interesse sind und welche nicht. Nur die interessanten Kollisionen werden dann von der Systemsoftware des Amiga 500 an die Programme gemeldet, die sich dafür interessieren. Wie die Programme darauf reagieren, ist deren Sache – in dieser Hinsicht bekommen Sie keine Unterstützung.

Trotzdem ist dieser »Service« der Kollisionserkennung (die man natürlich auch »abschalten« kann) eine sehr praktische Angelegenheit. Das wird jeder einsehen, der nur ein wenig Ahnung von Programmierung hat. Man muß sich dazu einmal klarmachen, wie schwierig es ist, mit einem Programm festzustellen, wann ein beliebig geformtes (!) Objekt mit einem anderen, das natürlich ebenfalls eine beliebige Form haben kann, zusammenstößt.

13 Klangerzeugung

Nicht ganz so große Publizität wie seine Grafikfähigkeiten haben die Möglichkeiten gefunden, die der Amiga 500 zur Klangerzeugung besitzt. Sie sind aber trotzdem auch recht beeindruckend. Der Amiga 500 besitzt insgesamt 4 voneinander unabhängige Tonkanäle, von denen jeweils zwei zu einem Stereokanal zusammengefaßt werden können. Jeder der 4 Kanäle kann eine völlig beliebige Wellenform mit beliebiger Hüllkurve ausgeben. Dazu wird nur in sehr geringem Maß ein Eingreifen des »zuständigen« Programms benötigt – der Amiga macht fast alles allein.

Um zu verstehen, wie ein Computer aber überhaupt dazu eingesetzt werden kann, Musik (und natürlich auch andere Geräusche) zu machen, müssen Sie zunächst einen kleinen Ausflug in die »digitale Musik« machen.

13.1 Töne, wie sie der Amiga sieht

Ein Ton oder Klang ist nichts anderes als eine Folge ständiger Luftdruckänderungen. Diese Luftdruckänderungen bewegen die menschlichen Trommelfelle. Die Hörnerven und das restliche Nervensystem des Menschen (vor allem auch das Gehirn) machen aus diesen Bewegungen des Trommelfells dann den Klang, den wir wahrnehmen. Ähnliche Luftdruckschwankungen muß der Amiga also zwangsläufig auch erzeugen, um einen (einigermaßen) natürlichen Klang zu simulieren. Da es aber Verstärker und Lautsprecher gibt, die aus Spannungsschwankungen Luftdruckschwankungen machen, reduziert sich das Problem für den Amiga 500 auf die Erzeugung einer im »richtigen Rhythmus« schwankenden Spannung an den beiden Lautsprecherausgängen.

Zur Erzeugung einer Klangform haben Sie zwei Alternativen. Sie können einen Klang entweder völlig »frei erfinden« oder einen natürlichen Klang digitalisieren. Zum Digitalisieren werden die Luftdruckschwankungen, die bei einem natürlichen Klang entstehen, in sehr kurzen Zeitabständen gemessen. Diesen Meßvorgang nennt man in der Musikbranche

üblicherweise »Sampling« oder »Sound-Sampling«. Entscheidend für die Qualität – also die »Natürlichkeit« – eines so digitalisierten Klangs ist die Anzahl der Meßwerte, die pro Sekunde aufgenommen werden. Dies nennt man die »Sampling-Frequenz« oder »Sampling-Rate«. Es läßt sich mathematisch nachweisen, daß für eine einigermaßen naturgetreue Digitalisierung eines Klangs eine Sampling-Frequenz nötig ist, die mindestens doppelt so hoch ist wie die höchsten Töne, die in dem Klang vorkommen.

Leider kann ein Computer aber Werte nicht beliebig genau messen und abspeichern wie zum Beispiel ein Kassettenrecorder. Sie erinnern sich vielleicht an das »digitale Prinzip« aus dem letzten Kapitel? Statt dessen ordnet er jedem gemessenen Wert eine Zahl zwischen einem ganz bestimmten Minimum und Maximum zu. Deswegen heißt dieser Vorgang auch »digitalisieren«. Aus einer kontinuierlichen (analogen) Schwingung wird eine (digitale) Folge von diskreten Zahlenwerten. Die folgende Abbildung zeigt in vereinfachter Form ein solches Sampling, bei dem jedem der gemessenen Werte eine Zahl zwischen 128 und +127 zugeordnet wird. Die dicke schwarze Linie stellt darin die Luftdruckschwankungen eines Tons dar, wie sie sich über die Zeit hinweg (die waagerechte Achse) ändern. Die gestrichelten Linien markieren die Zeitpunkte, an denen der Luftdruck jeweils gemessen wird.

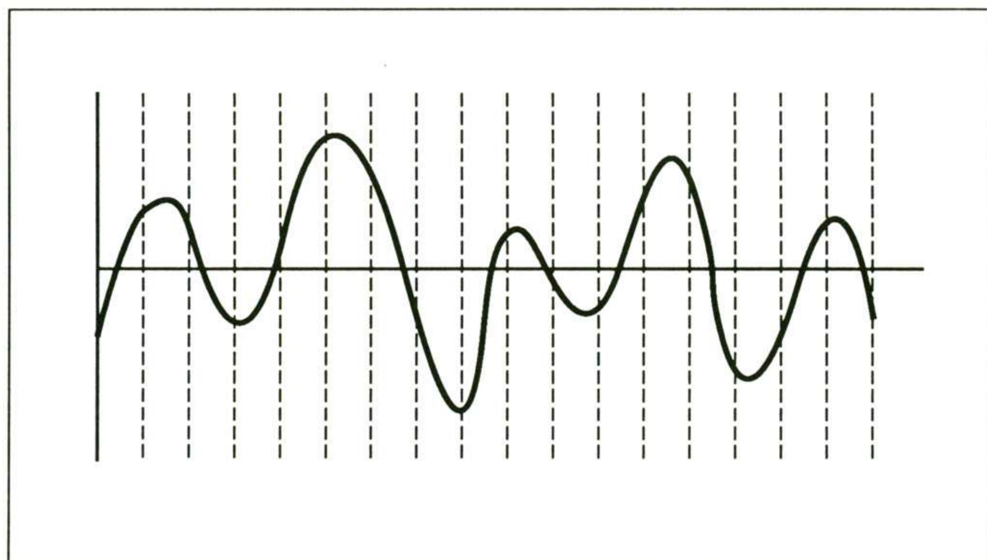


***Bild 13.1:** Ein Klang wird digitalisiert*

Ein solcher Meßvorgang (Sampling) ergibt eine Folge von Zahlenwerten, die dem Schwanken des Luftdrucks um die Nullachse entsprechen. Im Fall des Amiga 500 liegen diese Werte zwischen 128 und +127, was dem Fachmann natürlich sofort sagt, daß es sich dabei um 8-Bit-Zahlen (Bytes) handelt. Man spricht dann manchmal auch von Zahlen mit 8 Bit »Breite« oder von »8 Bit breitem Sampling«.

Obwohl ein solcher Zahlenwert nur sehr kurze Zeit gültig ist, »nimmt der Computer an«, daß er bis zum nächsten Meßpunkt unverändert bleibt. Etwas anderes kann er nach dem digitalen Prinzip auch gar nicht »annehmen«, da es für ihn keine feinere Aufteilung der Zeit zwischen zwei Meßpunkten gibt. Wenn ein digitalisierter Klang wieder ausgegeben wird, bleibt die Spannung (der Luftdruck) deshalb zwischen zwei Meßpunkten unverändert, woraus sich die typische Treppchenform eines digitalisierten Klangs ergibt.

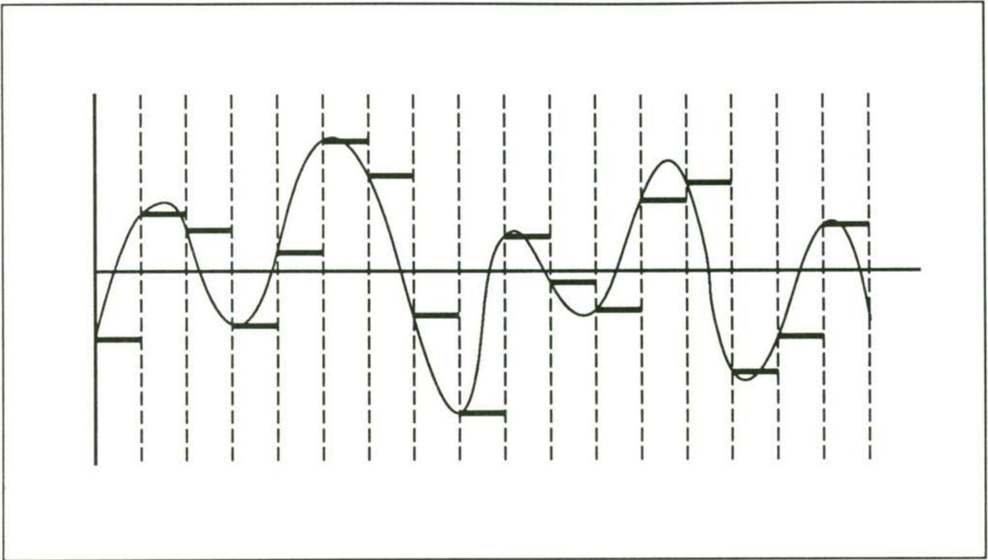


Bild 13.2: Ein digitalisierter Klang, wie er ausgegeben wird

Die dicken schwarzen Linien bezeichnen dabei wieder den Verlauf der vom Computer erzeugten Spannungsschwankungen, während die dünne Linie den Verlauf der Luftdruckschwankungen des »Originalklangs« vor der Digitalisierung beschreibt. Die gestrichelten Linien markieren die Zeitpunkte, zu denen die Spannung wechseln kann.

Diese Abbildung ist allerdings nur eine recht grobe Annäherung an die tatsächlichen Gegebenheiten. Die meisten klangerzeugenden Computer – so auch der Amiga 500 – enthalten elektronische Schaltkreise, die die plötzlichen Sprünge etwas glätten. Der digitalisierte kommt damit dem »echten« Klang wieder näher. Trotzdem sieht man hoffentlich, daß digitalisierte Klänge immer nur eine Annäherung an einen wirklichen Klang darstellen, die »der Natur« um so näher kommt, je kleiner die Abstände zwischen den einzelnen Messungen sind und je genauer der Wert gemessen wird.

Man darf von der Sound-Hardware des Amiga 500 deshalb nicht zuviel verlangen. Es handelt sich beim Amiga 500 nicht um einen Synthesizer, und seine Klangqualität kann auch zusammen mit einem guten Verstärker und guten Boxen HiFi-Tests gewiß nicht standhalten.

Für die Darstellung eines Klangs werden im Amiga 500 zum Beispiel nur 8 Bit breite Werte verwendet. HiFi-Geräte, die die Digitaltechnik nutzen (zum Beispiel CD-Plattenspieler) verwenden üblicherweise 16 Bit, was zu einer rund 250mal besseren Klangqualität führt – von anderen klangverbessernden Konstruktionen in diesen Geräten ganz abgesehen.

13.2 »Vollsynthetische« Klänge

Das eben beschriebene Digitalisierungsverfahren ist natürlich recht aufwendig. Man muß zum Beispiel einen natürlichen Klang mit einem Mikrofon aufnehmen – wobei eine gute Akustik des Aufnahmeraums sehr wichtig ist. Dies geht nur mit zusätzlicher Hardware, denn der Amiga 500 besitzt keinen Mikrofonanschluß. Aber auch mit dieser Zusatz-Hardware wird eine Digitalisierung selten auf Anhieb gelingen. Man muß statt dessen viel herumprobieren und mit der zugehörigen Digitalisierungs-Software sehr gut vertraut sein, bis man akzeptable Ergebnisse erzielen wird.

Einfacher hat man es da mit Klangformen, die man mathematisch beschreiben kann. Bei diesen »vollsynthetischen« Klängen kann man die Zahlenfolge, die den Luftdruckverlauf des Klangs beschreibt, nach einer – meist sehr einfachen – Formel berechnen. Die meistverwendeten dieser vollsynthetischen Klangformen sind die Sinus-, die Rechteck-, die Dreieck- und die Sägezahn-Welle.

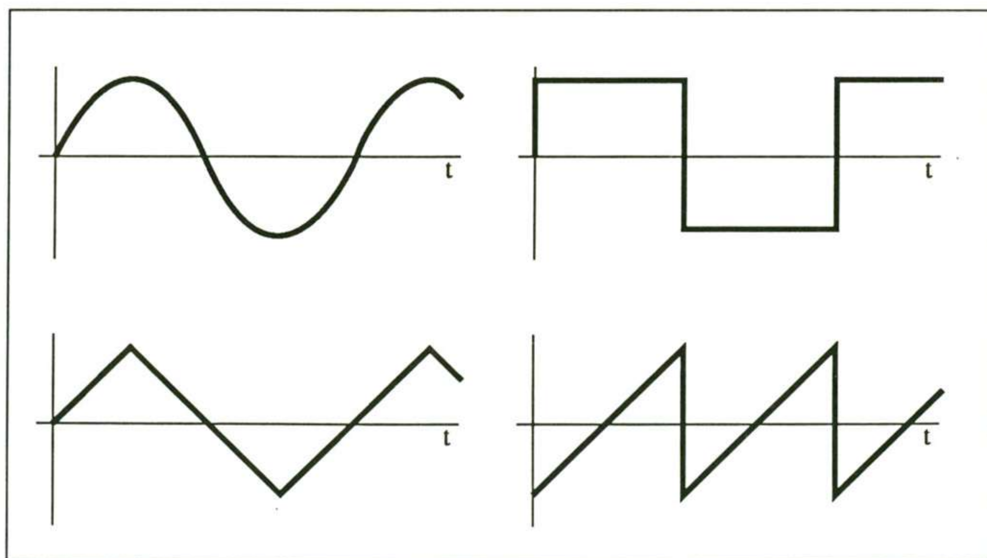


Bild 13.3: Die vier wichtigsten »vollsynthetischen« Klangformen

13.3 Die Hüllkurve eines Tons

Selbst bei der (verhältnismäßig) niedrigen Qualität der 8-Bit-Musik des Amiga 500 wäre es sehr aufwendig, mehrere Minuten Musik in der oben beschriebenen Form zu digitalisieren. Dabei spielt es keine Rolle, ob ein natürlicher Klang digitalisiert wird oder ein vollsynthetischer Klang berechnet wird. Jeder Wert der den Klang beschreibenden Zahlenfolge ist ja ein Byte (8 Bit). Bei einer Sampling-Frequenz von nur 10 kHz (10 000 Messungen pro Sekunden) fallen also jede Sekunde 10 Kbyte Daten an. In einer Minute sind das schon 600 Kbyte oder fast eine volle Diskette. Wenn Sie also gerne Ihre Lieblings-LP aus dem Computer hören würden, tun Sie gut daran, sich eine Festplatte anzuschaffen. Eine solche Festplatte mit 20 Mbyte Kapazität könnte gerade eine LP in digitalisierter Form aufnehmen.

Glücklicherweise ist eine vollständige Digitalisierung aber für viele Anwendungen gar nicht notwendig. Die meisten – zum Beispiel die von musikalischen Instrumenten erzeugten – Töne wiederholen sich nach kurzer Zeit, der sogenannten Periode. Die Länge dieser Periode liegt weit unterhalb einer Sekunde. Man braucht deshalb nur eine Periode zu digitalisieren und kann schon stundenlang einen Ton »halten«. Die Zahlenfolge mit den Luftdruckwerten, die sich aus der Digitalisierung ergibt, wird dazu einfach immer wieder hintereinandergehängt.

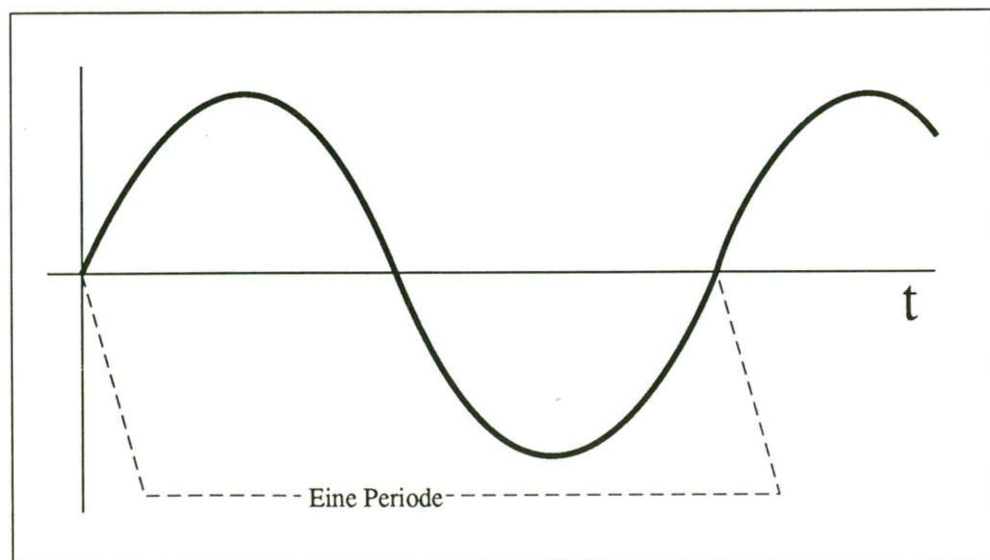


Bild 13.4: Die Periode eines Klangs

Hat man eine Periode eines Klangs erst einmal digitalisiert, kann man zudem auch die Tonhöhe (Frequenz) variieren. Dazu läßt man die Zahlenfolge der Luftdruckwerte einfach schneller oder langsamer ablaufen. Ohne für jede Tonhöhe zu einem bestimmten Instrument eine

Sampling-Folge zu haben, kann man dieses Instrument trotzdem in ganz verschiedenen Tonhöhen simulieren.

Um ein natürliches Instrument ausreichend zu charakterisieren, reicht aber die Klangform – also die Abfolge der Luftdruckschwankungen in einer Periode – nicht aus. Eine einzelne Note hat bei jedem Instrument typischerweise auch einen sehr charakteristischen Lautstärkenverlauf. Zu Beginn steigt sie sehr schnell zu einer maximalen Lautstärke an, fällt dann etwas ab, bleibt einen Moment auf einem Plateau nahezu gleicher Lautstärke stehen, um dann langsam auszuklingen. Bei manchen Instrumenten hört man das recht deutlich. Kirchenglocken zum Beispiel können noch Minuten nachklingen, wenn man sie einmal angestoßen hat.

Den typischen Verlauf der Lautstärke eines einmal angestoßenen Tons nennt man »Hüllkurve«, weil er die eigentlichen Schwingungen des Klangs einhüllt. Die Länge und Steilheit der verschiedenen Plateaus, Steigungen und Gefälle in der Hüllkurve ist von Instrument zu Instrument sehr verschieden und trägt entscheidend zu dem »typischen Klang« bei, der ein Instrument vom anderen unterscheidet.

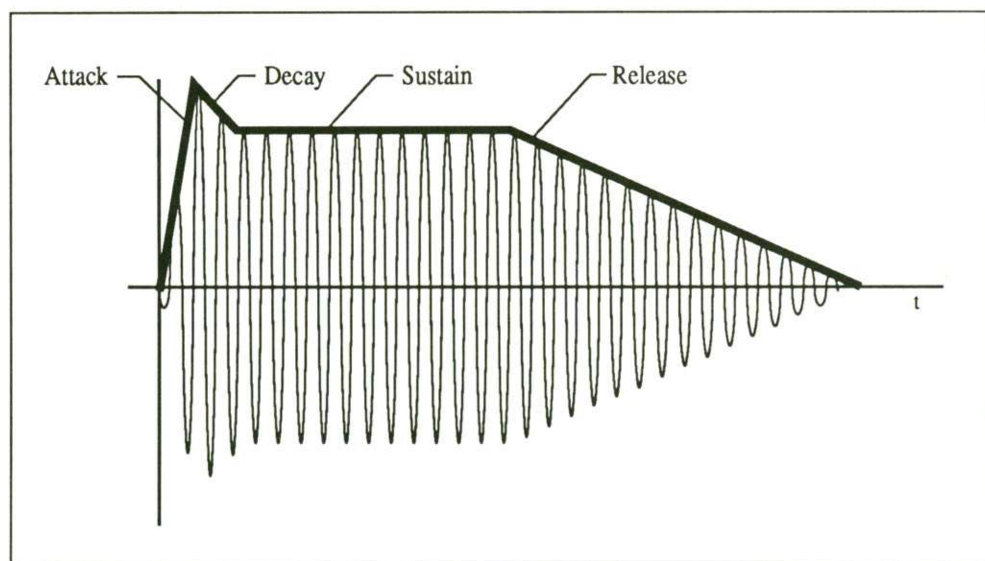


Bild 13.5: *Die Hüllkurve eines Tons*

Die Hüllkurve kann normalerweise schon durch sehr wenige Zahlenwerte genau beschrieben werden, und zwar durch drei Zeiten und drei Lautstärken. Dies sind die Zeiträume für das Erreichen der Maximal-Lautstärke, das Abfallen zum ersten »Plateau«, die Dauer des Plateaus, die Zeit, bis der Ton völlig verklungen ist und die Lautstärke, die der Ton jeweils am Ende jeder dieser Phasen hat. Der Fachmann nennt diese Zeiten *Attack*-(Anstiegs-), *Decay* (Abfalls-),

Sustain (Halte-) und *Release*-(Auskling-)Zeiten und -Levels. Die Hüllkurve wird nach den Anfangsbuchstaben dieser Namen manchmal auch ADSR-Kurve genannt.

13.4 Klangerzeugung mit dem Amiga 500

Nach dieser mehr theoretischen Vorrede über vom Computer erzeugte Klänge werden Sie nun aber auch erfahren, was man wirklich dazu benötigt, wenn der Amiga einen Ton erzeugen soll. Wie oben beschrieben, braucht man dazu vor allem eine digitalisierte Periode des gewünschten Klangs (also eine Zahlenfolge), die im Speicher hinterlegt werden muß. Diese Zahlenfolge kann man entweder bei einem wirklichen Ton messen oder mit Bleistift und Papier erstellen. Die Zahlenfolge für einen mathematisch beschreibbaren Klang, wie zum Beispiel die Sinuswelle, kann man zum Beispiel auch berechnen. In bestimmte Speicherstellen (für Fachleute: in Register des Soundchip Paula) schreibt man danach die Adresse, an der die Zahlenfolge im Speicher zu finden ist, ihre Länge und die Zeit, nach der jeweils ein neuer Wert der Zahlenfolge ausgegeben werden soll. Diese letzte Zahl bestimmt, wie schnell eine Periode ausgegeben wird und damit auch, welche Frequenz der erzeugte Ton hat.

Ist man mit diesen Vorarbeiten fertig, muß man in eine andere Speicherstelle einen speziellen Wert schreiben. Das ist das Signal für den Amiga (genauer: den Sound-Chip Paula), sich die Zahlenwerte für den Verlauf der Kurve zu holen und daraus Töne zu erzeugen. Wenn die Klangerzeugung erst einmal begonnen hat, braucht sich das Programm, das die Zahlenwerte für den Klang hinterlegt hat, nicht mehr darum zu kümmern. Der Amiga holt sich selbständig neue Werte aus der digitalisierten Zahlenfolge und wandelt diese in Spannungen am Lautsprecherausgang um.

Die weiter oben beschriebene Hüllkurve wird aber leider nicht durch die Hardware des Amiga 500 eingehalten. Hier muß die Systemsoftware helfend eingreifen und während der Erzeugung des Tons die Lautstärke regeln. Dies geht natürlich auf Kosten der Leistung des Prozessors, der sich in dieser Zeit nicht um andere Programme kümmern kann. Der Leistungsverlust fällt aber sehr klein aus, da diese Lautstärke-Änderungen immer nur nach Zeiträumen anfallen, die einem schnellen Mikroprozessor wie dem 68000 als halbe Ewigkeit erscheinen.

Leider läßt sich aber nicht jeder Klang durch eine kurze Periode beschreiben, die dann mit einer bestimmten Geschwindigkeit abgespielt werden kann. Viele Geräusche besitzen keine solche Regelmäßigkeit. In diesem Fall muß man entweder die Länge der digitalisierten »Periode« auf die gesamte Dauer des gewünschten Tons ausdehnen oder einen anderen Betriebsmodus der Sounderzeugung verwenden.

In diesem zweiten Modus macht die Hardware fast gar nichts mehr alleine. In diesem Modus muß das Programm die Spannung, die am Lautsprecherausgang anliegen soll, und die Dauer, während der sie konstant gehalten werden soll, direkt in bestimmte Speicherstellen schreiben. Ist die angegebene Zeitdauer vorbei, wird dies dem Programm gemeldet (für Fachleute: durch einen Interrupt) und ein neuer Wert verlangt. Das eigentliche Programm wird dadurch immer wieder (und zwar in sehr kurzen Abständen) unterbrochen und muß die Hardware mit neuen

Daten versorgen. Alle anderen Programme werden während einer so aufwendigen Klangerzeugung deshalb merklich langsamer.

13.5 Spezialeffekte

Bisher wurde nur beschrieben, wie ein Ton erzeugt wird. Der Amiga 500 besitzt aber vier Tonerzeugungs-Kanäle. Jeder dieser Kanäle kann völlig unabhängig von den anderen betrieben werden. Sie können von den vier verschiedenen Kanälen also – ähnlich wie mit vier verschiedenen Instrumenten – zur selben Zeit vier völlig verschiedene Töne erzeugen lassen. Verwendet ein Programm allerdings alle vier Kanäle im zweiten Betriebsmodus (siehe oben), verlangsamt das alle laufenden Programme schon recht beachtlich.

Die von den verschiedenen Kanälen erzeugten vier Stimmen werden auf zwei Lautsprecherausgänge verteilt. Der Programmierer kann bei der Tonerzeugung wählen, welcher Klang auf welchen Lautsprecherausgang geleitet wird. (Es müssen aber immer zwei Kanäle links und zwei rechts ausgegeben werden. Einen Kanal links und drei rechts auszugeben geht nicht.) Dadurch sind Stereoeffekte ohne weiteres möglich. Durch geschickte Programmierung können Sie also zum Beispiel einen Zug von links nach rechts fahren lassen.

Die Sound-Hardware des Amiga 500 erlaubt aber noch einige viel raffiniertere Tricks. Es können zum Beispiel zwei der Klangkanäle miteinander gekoppelt werden. Der eine (der sogenannte Oszillator) erzeugt dann den »eigentlichen« Ton, während die vom zweiten Kanal (dem sogenannten Modulator) erzeugte Schwingung zur Modulation des ersten Tons verwendet wird. Der Modulator besitzt dabei meist eine wesentlich niedrigere Frequenz als der Oszillator. Je nach der Einstellung der verschiedenen für diese Tricks zuständigen Register kann der Modulator nun die Frequenz des Oszillators oder seine Lautstärke ändern. Damit kann man die typischen Effekte erzielen, zu denen auch menschliche Musiker mit ihren Instrumenten fähig sind. Dazu gehören zum Beispiel Vibratos, Tremolos, Glissandos und so weiter.

Eine Kopplung zweier Kanäle für Spezialeffekte dieser Art findet übrigens in nahezu identischer Form auch in professionellen Synthesizern statt. Nur besitzen solche Synthesizer einige Klangkanäle mehr. Die Verwendung eines Kanals zur Modulation eines anderen schränkt deshalb die Anzahl der zur Verfügung stehenden Kanäle (Stimmen) kaum ein. Beim Amiga 500 hingegen muß man sich schon sehr genau überlegen, ob man diese Modulationsmöglichkeiten wirklich braucht oder lieber mehr Stimmen für eine Melodie verwenden möchte.

Obwohl die Möglichkeiten des Amiga 500 in dieser Hinsicht recht eingeschränkt sind, können Sie mit der passenden Software zumindest die Grundbegriffe der elektronischen Musikerzeugung kennenlernen. Das Programm Aegis Sonix zum Beispiel bietet am Bildschirm einige Regler und Einstellmöglichkeiten, die jedem, der schon einmal mit einem professionellen Synthesizer gearbeitet hat, vertraut sein werden. Wenn auch die Klangqualität des Amiga höchsten Ansprüchen nicht genügen wird, können Sie doch eine Menge damit lernen.

13.6 Sprachsynthese

Eine akustische Fähigkeit des Amiga 500 darf man auf keinen Fall vergessen: die Sprachausgabe. Der Amiga 500 (beziehungsweise sein Vorgänger, der Amiga 1000) dürfte so ziemlich der erste Computer sein, der ohne zusätzliche Software und Hardware sofort sprechen konnte. Die Hard- und Software, die zur Sprachsynthese benötigt wird, ist im Amiga 500 beziehungsweise im Betriebssystem »fest eingebaut«. Sprachsynthese kann deshalb in jedem Programm eingesetzt werden und ist problemlos zu programmieren. In der simpelsten Form übergibt man einfach einen ganz normalen Text an eine bestimmte System-Routine, die ihn daraufhin in akustischer Form ausgibt.

Die Aussprache kann man zusätzlich in verschiedenster Form beeinflussen. Die Stimmlage kann geändert werden. Der Amiga 500 kann entweder im tiefen Baß oder mit einer hellen Frauenstimme sprechen. Die Geschwindigkeit der Sprache kann innerhalb weiter Grenzen geändert werden – sogar in viel weiteren Grenzen als sinnvoll ist. Und schließlich kann der Amiga entweder sehr ausdrucksvoll, mit betonten Wörtern und Satzzeichen oder so monoton sprechen, wie es sich für einen Computer gehört.

So lesen Sie es zumindest in der Dokumentation des Amiga. Wenn Sie schon ein wenig mit dem Programm *Say* herumgespielt haben, das einen direkten Zugriff auf die Sprachsynthesefähigkeiten des Amiga 500 bietet, werden Sie schon wissen, daß die Sache nicht ganz so leicht ist – zumindest wenn Sie wirklich verständliche Sprache erzeugen wollen. Die Umsetzung von Buchstaben in zusammenhängend ausgesprochene Wörter und Sätze ist nämlich eine sehr komplexe Aufgabe, die durch viele Ausnahmeregeln erschwert wird. Überlegen Sie sich doch zum Beispiel einmal, wieviele verschiedene Möglichkeiten es gibt, einen Vokal wie *i* auszusprechen. Dieser eine Buchstabe kann völlig verschieden ausgesprochen werden, je nachdem, welche anderen Buchstaben sich in seiner »Nachbarschaft« befinden. Konsonanten und Wortbetonungen sind – zumindest für einen Computer – fast noch schwieriger zu handhaben.

Glücklicherweise kann man auch »eine Ebene tiefer« ansetzen. Die Sprachsynthese-Software des Amiga kann nicht nur Klartext aussprechen, sondern auf Wunsch auch »Phoneme«. Phoneme kennt jeder, der schon einmal in ein Wörterbuch geschaut hat. Es sind die Zeichen, mit denen die Aussprache von Wörtern beschrieben wird. Phoneme sind die »Urbausteine« der Sprache, elementare Silben und Laute, aus denen man jedes Wort (fast) jeder Sprache zusammensetzen kann.

Mit etwas Mühe – das heißt, bei manchen Wörtern müssen Sie die Phonemschreibweise im Wörterbuch nachschauen – kann man den Amiga auch dazu bringen, deutsche Sätze auszusprechen. Ein unverkennbarer (amerikanischer) Computer-Akzent bleibt allerdings. Trotzdem ist die Sprachausgabe für viele Anwendungszwecke – zum Beispiel für Fehlermeldungen oder zum Erschrecken von Freunden – recht brauchbar. Auch ohne auf den Bildschirm zu schauen, kann einem ein Programm auf diese Weise mehr Informationen mitteilen als mit einem kurzen Piepston, der sonst für solche Zwecke verwendet wird.

13.7 MIDI

Wie es inzwischen scheinbar schon guter Ton bei neuen Computern zu werden scheint, ist der Amiga natürlich auch für die Kommunikation mit MIDI-Geräten vorbereitet. MIDI (=*Musical Instruments Digital Interface*) ist der wohl wichtigste Standard für die Verbindung digitaler Musikinstrumente, wie Synthesizer und Keyboards. Hierzu benötigt man ein kleines Kästchen namens MIDI-Interface, das an den seriellen Ausgang des Amiga 500 angeschlossen wird und die Signale des Amiga dem MIDI-Standard anpaßt. Dann braucht man »nur noch« einen guten Synthesizer und ein MIDI-fähiges Keyboard.

Wem also die Klangqualität seines Amiga 500 auch dann noch nicht ausreicht, wenn er ihn an seine Stereoanlage angeschlossen hat, der sollte sich die Anschaffung des MIDI-Interfaces und der entsprechenden MIDI-Geräte überlegen.

Man mag sich nun vielleicht fragen, wozu der Computer überhaupt noch gut ist, wenn die Musik auf einer separaten Tastatur eingegeben und auf einem Synthesizer erzeugt wird. Die Antwort darauf ist wieder einmal mehr bei der Software zu finden. Es ist nämlich wesentlich komfortabler, Musik auf einem Computer zu entwickeln als auf einem Blatt Papier. Dies wird noch deutlicher, wenn Musik mit noch mehr als nur vier Stimmen erzeugt werden soll und die Bedienung eines leistungsfähigen (und komplizierten) Synthesizers durch den Musiker nötig wird.

Man könnte den Amiga 500 deshalb sehr sinnvoll als Oberaufseher in einem System MIDI-fähiger Komponenten einsetzen. Auf seinem Bildschirm bekommt man einen Überblick über die verschiedenen Vorgänge und Verbindungen gezeigt, man komponiert aber auch gleichzeitig auf dem Computer und entwickelt neue Klänge. Die Klangformen, Hüllkurven und Noten kann man sich natürlich direkt auf dem Amiga 500 probeweise anhören. »Gespielt« werden die Melodien dann aber auf einem Synthesizer, zu dem man die dazu nötigen Informationen über MIDI-Kabel schickt. Mit einem speziellen Keyboard kann man aber auch »direkt« den Synthesizer bedienen, ohne die Noten dazu erst in den Amiga eingeben zu müssen. Nur die Klangformen und Hüllkurven werden in diesem Fall vom Amiga geliefert. Das Programm *Deluxe Music Construction Set* der Firma *Electronic Arts* bietet zum Beispiel exakt diese Möglichkeiten!

14 Eine Handvoll Chips

In diesem Kapitel werden wir noch ein wenig tiefer in die »Innenwelt« des Amiga einsteigen. Nachdem Sie in den letzten beiden Kapiteln schon Grundsätzliches zu den Themen Grafik und Sound auf dem Amiga 500 erfahren haben, werden Sie nun einige wichtige Chips der Amiga-Hardware etwas näher kennenlernen. Keine Angst: Sie benötigen diese Informationen nicht zur Arbeit mit dem Amiga. Glücklicherweise gibt es heute fast keine Computer mehr, über deren Hardware oder Systemsoftware man detailliert Bescheid wissen muß, um damit umgehen zu können.

In diesem Kapitel geht es vor allem um die speziellen Chips, die – mehr als alle anderen Komponenten des Amiga-Systems – den Amiga 500 zum »Amiga« machen. Diese Chips geben dem Amiga 500 einen Großteil seiner besonderen Fähigkeiten. Sie tragen die Namen »Paula«, »Denise« und »Agnus«. (Es gibt noch einen Spezialchip namens »Gary«, der in diesem Zusammenhang aber nicht so interessant ist, da er nur die Funktionen vieler kleinerer Chips zusammenfaßt und ansonsten nicht viel Weltbewegendes leistet.) Diese liebevollen Bezeichnungen für kleine schwarze Chips mit stacheligen Beinen lassen erahnen, wie sie den Entwicklern am Herzen liegen.

14.1 Die CPU – Herr im Hause Amiga

Objektiv betrachtet ist sicherlich die sogenannte »CPU« (*Central Processing Unit* oder auf gut deutsch *Zentraleinheit*) der wichtigste Chip im Innern des Amiga 500, wenn auch die anderen Chips spektakulärer sind. Die CPU ist der Chip in jedem Computer, der den Löwenanteil der Arbeit zu leisten hat. Es ist dieser Chip, der Programme wirklich ausführt – also zum Beispiel rechnet und Entscheidungen trifft. Man könnte die CPU auch als das »Herz des Computers« bezeichnen.

Die CPU des Amiga ist der 16/32-Bit-Prozessor M68000 der Firma Motorola. Er wird auch in anderen Computern dieser Preisklasse sowie in professionellen Systemen für 30 000 DM und

mehr eingesetzt. Die Bezeichnung 16/32-Bit-Prozessor kommt daher, daß man sich nicht darüber einig werden kann, nach welchen Kriterien man einen Prozessor in welche Kategorie einordnet. Typischerweise ist es so, daß Prozessoren, die jeweils 8 Bit in einem Arbeitsschritt verarbeiten können (zum Beispiel zwei 8-Bit-Zahlen addieren können) 8-Bit-Prozessoren heißen. Der 6502 im Apple //e und im Commodore C64 fallen zum Beispiel in diese Kategorie. Prozessoren, die in einem Schritt 16 oder 32 Bit verarbeiten können, heißen entsprechend 16- oder 32-Bit-Prozessoren. Ein anderes Kriterium ist aber auch, wieviele Bits in einem Schritt aus dem Speicher gelesen oder in ihm abgelegt werden können. Der M68000 erfüllt bei einigen dieser Kriterien die Bedingungen für einen 32-Bit-Prozessor und bei anderen nur die für einen 16-Bit-Prozessor. (Ein Nachfolgemodell des 68000, der 68020 ist in jeder Hinsicht ein echter 32-Bit-Chip. Er wird unter anderem im Apple Macintosh II eingesetzt und kann auch im Amiga 2000 verwendet werden.)

Kommen wir nun aber zu den drei Spezial-Chips. Bei ihnen handelt es sich um sogenannte »Custom-Chips«, das heißt, sie wurden speziell für einen Computer-Hersteller (in diesem Falle Commodore beziehungsweise Amiga) entwickelt und werden auch nur für ihn hergestellt. Die meisten anderen Chips, die Sie im Amiga 500 finden werden, sind Standard-Bauteile, die jeder Kunde dem Chip-Hersteller abkaufen und dann in seinen eigenen Produkten verwenden kann. Standard-Chips sind fast immer billiger als Custom-Chips, weil sie üblicherweise in größeren Stückzahlen verkauft werden. Sie stellen andererseits aber auch immer einen Kompromiß zwischen den Wünschen vieler Anwender dar. Für wirklich einzigartige Möglichkeiten oder besonders hohe Leistungen benötigt man deshalb Custom-Chips.

14.2 Teamwork

Die Hardware des Amiga 500 betreibt echtes Teamwork. Die drei Zusatz-Chips »Agnus«, »Paula« und »Denise« arbeiten unabhängig von der CPU und gleichzeitig mit. Das ist eine Behauptung, die Sie vielleicht schon einmal gehört haben, deren Bedeutung aber nicht immer sofort klar ist. Der Vorteil, den dieses Arbeitsprinzip mit sich bringt, ist auch nicht ganz so leicht zu erklären, weshalb ich etwas weiter ausholen werde.

14.2.1 Koprozessoren

Es gibt viele Computer, die alle anfallenden Arbeiten der CPU überlassen. Die CPU muß Grafikdarstellungen, Musik, Berechnungen und natürlich auch die Ein- und Ausgabe übernehmen. Da alle Arbeiten aber von einem Prozessor übernommen werden, müssen diese Tätigkeiten hintereinander ausgeführt werden. Jede dieser Aufgaben hält die CPU also eine Weile von den anderen Aufgaben ab.

Die Spezial-Chips des Amiga nehmen nun dem Hauptprozessor Arbeit ab. Grafikdarstellungen, Sounderzeugung und noch andere Arbeiten werden durch die drei Chips erledigt. Der Prozessor hat dann zwar immer noch etwas – aber nicht mehr sehr viel mit diesen Aufgaben zu tun. Um diese Verlagerung von Aufgaben möglichst effektiv zu gestalten, können manche Spezial-Chips ihre Arbeit gleichzeitig mit der CPU erledigen. Es handelt sich dann um

sogenannte »Koprozessoren«, die völlig unabhängig von der CPU ihrer Arbeit nachgehen. Die CPU bleibt natürlich immer noch »Herr im Hause Amiga«, gibt aber – wie ein guter Manager – »niedere Aufgaben« an ihre Mitarbeiter weiter und wendet sich dann anderen Dingen zu.

Man muß sich das so vorstellen, daß die CPU eine Aufgabe vorbereitet und den Koprozessor mit allen nötigen Informationen darüber versorgt. Dann sagt sie dem Koprozessor, daß er mit der Arbeit beginnen soll und kümmert sich um andere Dinge. Währenddessen wird auch der Koprozessor tätig und erledigt die ihm aufgetragene Arbeit. Wenn er damit fertig ist, signalisiert er das der CPU (mit einem sogenannten Interrupt) und kann wieder neu mit Arbeit versorgt werden oder eine Pause einlegen (auch Chips brauchen mal eine Pause). Statt sich vom Koprozessor die Beendigung einer Arbeit mitteilen zu lassen, kann die CPU aber auch »zwischendurch immer mal wieder nachschauen«, ob der Koprozessor schon mit seiner Arbeit fertig ist und gegebenenfalls darauf reagieren. (Der Computerfachmann nennt diesen Vorgang des regelmäßigen Nachschauens übrigens »Polling«.)

14.2.2 DMA

Bei einer solchen Konfiguration aus CPU und Koprozessoren tritt oft nur ein Problem auf. Sowohl die CPU wie auch die Koprozessoren besitzen üblicherweise nur sehr wenig eigenen Speicherplatz, die sogenannten »Register«. Für größere Aufgaben benötigen sie deshalb den RAM-Speicher, der ungleich mehr Daten fassen kann. In vielen Computern gibt es Koprozessoren und andere Hilfs-Chips, die Daten nicht selbst aus dem Speicher holen oder dort hinterlegen können, sondern dazu immer die CPU benötigen, weil diese als einziger Chip an den Speicher angeschlossen ist. Immer, wenn solche »unselbständigen« Chips Daten brauchen oder im Speicher ablegen wollen, müssen Sie die CPU von ihren eigentlichen Aufgaben abhalten und sie als »Datenschaufel« mißbrauchen. Ein paralleles Arbeiten, das nötig ist um einen Zusatz-Chip so richtig auszunutzen, ist also nur so lange möglich, wie der Speicher nicht benötigt wird.

Auf dem Amiga ist das anders! Hier sind nämlich auch die Koprozessoren selbst an den Speicher angeschlossen. Das entsprechende Arbeitsprinzip heißt »DMA« (für *direct memory access* oder auf deutsch *direkter Speicherzugriff*). Von Chips, die von selbst auf den Speicher zugreifen können, spricht man deshalb auch als DMA-Chips. Auf den Speicher des Amiga kann man über insgesamt 25 Wege oder »Kanäle« zugreifen. Einige dieser DMA-Kanäle nutzen auch die Spezial-Chips, andere sind für künftige Erweiterungen noch frei.

DMA ist ein recht fortschrittliches und teures Konzept, das erst vor relativ kurzer Zeit Eingang in preiswerte Computer gefunden hat. Es ist für viele Anwendungen nützlich. Neben Aufgaben wie Grafik und Klangerzeugung – die DMA benötigen, weil bei diesen Vorgängen Daten aus dem Speicher gelesen und in Bilder und Töne umgewandelt werden müssen – gibt es noch eine Reihe anderer Anwendungen für DMA. Dazu gehören vor allem auch die Ein- und Ausgabe von Daten an Peripheriegeräte wie Bildschirm, Drucker und so weiter.

Die DMA-Chips des Amiga 500 nehmen dem Hauptprozessor aber nicht nur Arbeit ab. Sie können die ihnen übertragenen Arbeiten auch viel schneller erledigen als der Hauptprozessor das überhaupt könnte. Die CPU ist schließlich nicht speziell für Grafikanwendungen oder für

das Hin- und Herschieben von Daten im Speicher gebaut worden – wie die Spezial-Chips es sind.

14.2.3 Probleme und Lösungen

Obwohl die Spezial-Chips die CPU nicht für den Speicherzugriff benötigen, tritt trotzdem ein Engpaß beim »Eingang in den Speicher« auf. Die Chips und der Prozessor können normalerweise nämlich nicht gleichzeitig auf den Speicher zugreifen. Dieser kann nämlich immer nur mit einem anderen Bauteil zur gleichen Zeit kommunizieren. Es gibt zwar besondere Speicher, die mit zwei oder mehr anderen Chips gleichzeitig Daten austauschen können (*dual ported RAMs*), diese sind aber sehr viel teurer als die im Amiga verwendeten Chips und für einen Personalcomputer wie den Amiga somit kaum geeignet.

Glücklicherweise kann der Speicher jedoch mit viel höherer Geschwindigkeit Daten aufnehmen und abgeben, als die CPU dies kann. Diese Geschwindigkeit, mit der Chips arbeiten können, nennt man ihre »Taktfrequenz« und ist gleich der Anzahl von »Taktzyklen«, die in einer Sekunde durchlaufen werden. Die CPU des Amiga durchläuft in jeder Sekunde circa 7,2 Millionen solcher Taktzyklen und hat somit eine Taktfrequenz von 7,2 MHz (Megahertz). In jedem Taktzyklus kann die CPU einen Arbeitsschritt durchführen, also zum Beispiel ein Wort (16 Bit) Daten vom Speicher empfangen. Der RAM-Speicher läuft aber mit einer Taktfrequenz von 14,4 MHz, hat also in der Sekunde die doppelte Anzahl von Taktzyklen und kann deshalb doppelt so viele Daten in jeder Zeiteinheit liefern als die CPU empfangen kann.

Diesen »Geschwindigkeitsvorsprung« des Speichers haben die Entwickler des Amiga ausgenutzt, indem sie den RAM-Speicher nur jeden zweiten Takt mit der CPU kommunizieren lassen. Alle anderen Takte werden auf die restlichen DMA-Chips verteilt. Da diese den Speicher meist nicht so intensiv benötigen wie die CPU, können so CPU und Spezial-Chips ungestört voneinander und gleichzeitig ihrer Arbeit nachgehen.

Nur unter ganz besonderen Umständen benötigen die Spezial-Chips mehr Taktzyklen des Speichers. In diesen Fällen wird die CPU kurz angehalten, wenn Sie den Speicher zur selben Zeit benötigt wie ein anderer Chip und darf wieder weitermachen, wenn dieser Chip den Speicher nicht mehr braucht. (»Kurz« heißt in diesem Fall wirklich extrem kurz. Es handelt sich dabei um Zeitspannen, in denen kein Mensch reagieren oder auch nur etwas von einer solchen Pause merken könnte.) Diese Fälle, in denen die CPU gebremst wird, kommen aber nur sehr selten vor. Wenn Sie zum Beispiel 32 verschiedene Farben gleichzeitig am Bildschirm sehen wollen, bremst Denise (der Videochip) die CPU merklich ab, weil er wesentlich mehr Daten aus dem Speicher holen muß, um die ganze Farbenpracht darzustellen. Wenn Sie sich aber mit weniger Farben begnügen, üben die CPU und die Spezial-Chips friedliche Koexistenz – Gorbatschow läßt grüßen.

Damit sollten Sie jetzt zumindest einen ersten Eindruck vom internen Aufbau des Amiga und der modernen arbeitsteiligen Zusammenarbeit seiner Komponenten erhalten haben. Kommen wir nun zu den Chips im einzelnen.

14.3 Paula, die Musikalische

Einer der drei Spezialchips des Amiga trägt den Namen »Paula«. Das ist nicht nur irgendein Name, sondern leitet sich auf eine heute nicht mehr ganz nachvollziehbare Weise aus den englischen Worten *peripheral* und *audio* ab, was schon andeutet, wofür dieser Chip zuständig ist. Er kontrolliert die Arbeit mit den Peripheriebausteinen und ist für die Klangerzeugung zuständig. Und er kann noch mehr, denn er regelt die sogenannte »Interruptverarbeitung«.

14.3.1 Interrupts

Interrupts sind, wörtlich übersetzt, Unterbrechungen. Dieser Ausdruck bezieht sich meist darauf, daß andere Chips oder externe Geräte die CPU in ihrer Arbeit unterbrechen, weil irgendein Ereignis eingetreten ist, das die Aufmerksamkeit der CPU verlangt. Koprozessoren senden zum Beispiel Interrupts an die CPU, wenn sie mit ihrer Arbeit fertig sind, und Ein-/Ausgabe-Bausteine schicken Interrupts, wenn Daten bereitstehen und die CPU entscheiden muß, was damit gemacht werden soll. Es gibt verschiedene Interrupts, mit denen der CPU verschiedene Nachrichten geschickt werden können. Zusätzlich stellen die Chips, die Interrupts verursachen, meist aber auch noch weitere Informationen in bestimmten Registern bereit, aus denen die CPU mehr über die näheren Umstände eines Interrupts entnehmen kann.

Bei einem Interrupt wird der gesamte Zustand des gerade laufenden Programms schnappschußartig festgehalten und ein spezielles kleines Programm gestartet, das sich um die Behandlung des Interrupts kümmert. Der Fachmann nennt solche Programme »Interrupt-Handler«. Ist der Interrupt-Handler fertig, wird das unterbrochene Programm wieder fortgesetzt, wobei zuvor der alte Programmzustand mit Hilfe des gespeicherten Schnappschusses wiederhergestellt wird.

Verschiedene Arten von Interrupts besitzen zudem auch verschiedene Prioritäten, die besagen, wie wichtig die Unterbrechung ist. Es kann ja sein, daß gerade kein »normales Programm« läuft, wenn ein Interrupt eintritt, sondern ein Interrupt-Handler. Falls dieser »wichtiger« ist als der neu eingetroffene Interrupt, darf er auf keinen Fall unterbrochen werden. Genau dies ermöglichen die Prioritäten. Es gilt die Regel, daß Interrupts niedrigerer Priorität keine Interrupt-Handler von gleicher oder höherer Priorität unterbrechen können.

Der Paula-Chip hilft nun der CPU dabei, die Interrupts der verschiedenen Systemkomponenten zu erkennen und richtig einzuordnen. Das AmigaDOS nutzt diese Interruptverarbeitung auch aus, um das Multitasking zu ermöglichen. Hierzu wird in regelmäßigen Abständen ein spezieller Interrupt erzeugt, der dazu führt, daß das gerade laufende Programm unterbrochen wird. Nach dem Ende des entsprechenden Interrupt-Handlers wird aber das unterbrochene Programm nicht wieder aufgenommen, sondern ein anderes, das bislang in Wartestellung stand, wird aktiv. Kurz darauf erfolgt wieder ein Interrupt, woraufhin das nächste Programm eine Weile arbeiten darf, bis es auch wieder unterbrochen wird. Irgendwann einmal – je nachdem, wieviele Programme arbeiten wollen – ist auch das zuerst unterbrochene Programm dann wieder »an der Reihe«.

Paula hilft der CPU in folgender Weise, Interrupts einfacher handhaben zu können: sie empfängt (noch vor der CPU) alle Interrupts der für Ein-/Ausgabe zuständigen Chips und der verschiedenen DMA-Chips und speichert alle Informationen darüber in speziellen Registern. In diesen Registern kann die CPU später nachschauen, was bei einer »Unterbrechung« genau vorgefallen ist. Paula gliedert die vielen möglichen Interrupts dabei zugleich auch in unterschiedliche Gruppen auf. Jede dieser Gruppen vereinigt Interrupts ähnlicher Art und gleicher Priorität. Da Paula alle diese Vorarbeiten schon leistet, bevor die CPU etwas von einem Interrupt erfährt, braucht diese viel weniger Zeit, um einen Interrupt einzuordnen und zu entscheiden, wie »angemessen« darauf zu reagieren ist.

14.3.2 Ein- und Ausgabe

Eine weitere Aufgabe des Paula-Chips ist der Kontakt zwischen Außenwelt und System. Paula betreut die Ein-/Ausgabe-Bausteine, mit deren Hilfe die verschiedenen Schnittstellen angesteuert werden. Dazu gehören die parallele, die serielle Schnittstelle, der Anschluß für ein externes Diskettenlaufwerk und die beiden Maus-Anschlüsse. Daß Paula, also ein von der CPU getrennter Chip, diese Aufgabe übernimmt, bedeutet wieder eine Entlastung für die CPU, die sich sonst selbst um diese Aufgaben kümmern müßte. Paula dient sozusagen als Mittelsmann (Mittelsfrau?) zwischen dem eigentlichen Computer und den an den verschiedenen Schnittstellen angeschlossenen externen Geräten.

14.3.3 Klangerzeugung

Auch die Klangerzeugung und Klangausgabe auf dem Amiga ist (nahezu) ausschließlich eine Sache des Paula-Chips. Die Schaltungen, die die Informationen über Form, Lautstärke und Dauer eines Klangs in Spannungsschwankungen umsetzen, aus denen ein Lautsprecher dann die Töne macht, befinden sich alle in Paula. Das Prinzip der elektronischen Klangerzeugung auf dem Amiga wurde bereits in Kapitel 13 beschrieben.

14.4 Denise das Grafik-Wunder

Während Paula für die Klangerzeugung zuständig ist, erledigt der Custom-Chip Denise den größten Teil der bei der Erzeugung von Bildern anfallenden Aufgaben. Der Name »Denise« hat ähnlich mysteriöse Ursprünge wie der Name Paula und stammt von der Bezeichnung *Display Encoder* (zu deutsch etwa *Bild-Kodierer*) ab. Dieser Name deutet schon darauf hin, daß Denise hauptsächlich für die Erzeugung des Bildsignals zuständig ist, das an den Monitor beziehungsweise Fernsehbildschirm geht. Beim eigentlichen Aufbau des Bildes im RAM-Speicher ist ein anderer Chip behilflich, Agnus, der in einem späteren Abschnitt noch näher beschrieben wird.

14.4.1 Grafik und Farbe

Wenn Sie sich noch einmal Kapitel 12 in das Gedächtnis zurückrufen, werden Sie sich vielleicht daran erinnern, was eine *Bitmap* ist. In Form einer Bitmap werden im RAM-Speicher die Bilder abgespeichert, die hinterher am Bildschirm erscheinen. Zu jedem Punkt am

Bildschirm enthält die Bitmap die Information, welche Farbe dieser Punkt hat. Denise enthält die Schaltungen, die den Inhalt der Bitmap lesen und die Farbinformationen in elektrische Signale umsetzen. Diese Signale gehen dann über das Monitorkabel zu Ihrem Monitor und steuern dort den Elektronenstrahl, der das Bild auf die Bildröhre zeichnet. Hierzu liest Denise die Informationen zu den einzelnen Bildpunkten in genau derselben Reihenfolge aus der Bitmap im Speicher, in der auch der Elektronenstrahl die Bildpunkte auf die Bildröhre zeichnet.

Das Prinzip, nach dem eine Bitmap aufgebaut ist und die Farben darin abgespeichert sind, wurde bereits in Kapitel 12 beschrieben.

14.4.2 Bewegung ins Bild!

Die eben beschriebene Umsetzung einer Bitmap in eine Bildschirm-Grafik ist statisch – also unbeweglich. Wenn irgendetwas an dem Bild am Monitor geändert werden soll, muß dafür die Bitmap geändert werden. Denise kennt aber auch noch zweierlei Arten bewegter Grafiken. Zum einen kann der Hintergrund einer Grafik (das sogenannte Playfield) in Wirklichkeit eine sehr große Fläche, größer als der Bildschirm, sein, aus dem nur ein Teil gezeigt wird. Dieser sichtbare Ausschnitt kann stufenlos und schnell über dem gesamten Playfield verschoben werden. Die zweite Möglichkeit, Bewegung ins Bild zu bringen, sind die sogenannten Sprites, kleine farbige Bilder, die sich »über« dem Playfield befinden und sehr schnell, unabhängig von diesem, bewegt werden können. Sprites erfreuen sich auch auf anderen Computern schon großer Beliebtheit und lassen sich (gegenteilig lautenden Meinungen zum Trotz) auch außerhalb von Video-Spielen sinnvoll einsetzen. Auch der Mauszeiger ist zum Beispiel ein Sprite.

Der Denise-Chip erkennt außerdem noch Kollisionen verschiedener Sprites mit Teilen der Hintergrundgrafik und mit anderen Sprites. Programme können diese Kollisionsabfrage benutzen und eine entsprechende Reaktion darauf einleiten.

14.5 Agnus, der Rechenkünstler

»Agnus« ist eine Abkürzung, die auf heute unerfindlichen Wegen aus »Adreßgenerator« entstanden ist. Agnus beinhaltet aber noch mehr als nur diesen Adreßgenerator. Weitere wichtige Bestandteile dieses Chips sind der »Blitter« und der »Copper«, über die Sie in diesem Abschnitt noch mehr erfahren werden. Eigentlich heißt der Agnus-Chip im Amiga 500 übrigens »Fat-Agnus« (dicker Agnus), weil er gegenüber dem ersten Agnus im Amiga 1000 noch eine Menge Funktionen dazubekommen hat. Eine solch unfreundliche Bezeichnung möchte ich diesem flinken Rechenkünstler aber ersparen. Agnus ist alles andere als fett und träge.

14.5.1 Was ist ein Adreßgenerator?

Ein Hauptbestandteil des Agnus-Chips ist ein großer Adreßgenerator. Die Custom-Chips des Amiga haben ja unabhängig von der CPU direkten Zugriff auf den RAM-Speicher. Dazu

benötigen diese Chips jedoch sogenannte »Adreßgeneratoren« und DMA-Kanäle. Über DMA-Kanäle wird der Zugriff auf den RAM-Speicher abgewickelt. Ein Adreßgenerator wandelt die Signale eines Chips in vollständige Speicheradressen um oder ordnet bestimmte Speicheradressen einzelnen Registern in den Chips zu. Man kann sich das so vorstellen, daß von einem Haus statt der Postanschrift nur die Beschreibung »erste Straße nach dem Ortseingang und dann drittes Haus auf der linken Seite« bekannt ist. Ein Adreßgenerator würde daraus dann »Schillerstraße 12, 4783 Klotzhausen« machen.

Keiner der beiden anderen Chips enthält jedoch alle DMA-Kanäle oder Adreßgeneratoren. Alle Speicherzugriffe laufen deshalb über Agnus und werden von ihm kontrolliert und koordiniert. Insgesamt hat Agnus 25 DMA-Kanäle, über die 25 verschiedene Geräte ohne Beihilfe der CPU Daten in den Speicher schreiben oder aus ihm lesen können.

14.5.2 Der Blitter

Eine der wichtigsten Vorgaben bei der Konstruktion des Amiga war es, eine möglichst schnelle Erzeugung und Änderung von Grafiken zu ermöglichen. Insbesondere sollten ohne viel Aufwand »animierte« bewegte Grafiken möglich sein. Zum Teil werden diese Aufgaben von Denise erledigt. Sprites ermöglichen zum Beispiel sehr schnelle Bewegungen auf dem Bildschirm. Sprites sind von ihren Möglichkeiten her aber auch sehr eingeschränkt (diese Einschränkungen wurden in Kapitel 12 ausführlich beschrieben). Für aufwendigere Effekte muß deshalb die Bitmap, die einem Bild zugrunde liegt, geändert werden. Im Gegensatz zur Änderung von Sprites müssen zur Änderung der Hintergrund-Bitmap eines Bildes große Mengen von Daten blitzschnell im Speicher verschoben werden. Dies ist die Spezialität des »Blitters«, eines relativ selbständigen Prozessors innerhalb von Agnus.

Der Name »Blitter« hat schon eine recht lange Tradition und beruht auf der Prozedur »BitBlt« (***bit block transfer***), die es auf einem älteren Computer der Firma XEROX gab. Diese Prozedur war für das schnelle Verschieben »rechteckiger« Bereiche in einer Bitmap zuständig. Und da die für den Amiga entwickelte Hardware-Einheit innerhalb von Agnus genau dasselbe tut, lag es nahe, diesen Namen zu verwenden. Jay Miner, der Designer der drei Amiga-Chips, ist in dieser Hinsicht anderer Meinung. Der Blitter des Amiga kann auch wirklich einiges mehr, als nur Daten hin- und herzuschieben. Jay Miner nennt ihn deshalb Bimmer für *Bit Image Manipulator* (zu deutsch etwa *Bit-Bild-Manipulierer*). Dies soll andeuten, daß dieser Prozessor nicht nur Daten durch den Speicher schaufeln, sondern zugleich komplexe Operationen damit durchführen kann.

Der Blitter hat in vielen Grafik-intensiven Programmen eine ganze Reihe von Aufgaben zu übernehmen. Die erste davon ist das schnelle Füllen von Flächen (wie es zum Beispiel Graphicraft und Deluxe Paint demonstrieren). Der Blitter kann Flächen mit einer Geschwindigkeit von einer Million Bildpunkten pro Sekunde füllen. Mit derselben Geschwindigkeit kann er auch gerade Linien ziehen. Beide Fähigkeiten verknüpft er übrigens auch sehr geschickt. Um eine bestimmte Fläche auf dem Bildschirm zu füllen, muß diese durch mindestens eine dünne Linie in einer anderen Farbe begrenzt sein. Programme zeichnen polygonförmige Flächen (zum Beispiel auch Rechtecke) deshalb, indem Sie den Blitter blitzschnell die Seiten des Polygons ziehen lassen und dann das Innere wieder durch den Blitter

füllen lassen. Dieser Vorgang läuft so schnell ab, daß die Fläche scheinbar schlagartig auf dem Bildschirm erscheint.

Wichtigste Aufgabe des Blitters ist aber die Bewegung von Objekten auf dem Bildschirm. Sinnigerweise heißen diese Objekte dann auch ***Blitter Objects*** (BOBs). Der Blitter kann extrem schnell (so schnell, wie es die Speicherchips überhaupt nur können) Daten im Speicher von einem Ort zu einem anderen bewegen. Um BOBs als eine Art eigenständiger Bilder »über der Hintergrundgrafik« zu bewegen, werden diese mit Hilfe des Blitter direkt in die Bitmap des Hintergrundbildes hineinkopiert. Vorher wird (ebenfalls mit dem Blitter) der alte Inhalt der Bitmap in einem Puffer gesichert. Wird das BOB dann bewegt, so wird diese gesicherte Kopie des alten Inhalts in die Bitmap zurückkopiert, der Inhalt des Hintergrundbildes an der neuen Position des BOBs gesichert, und dann das BOB an der neuen Stelle gezeichnet. Für diese verschiedenen Aktionen wird zwar auch viel geschickt programmierte Software benötigt, den Löwenanteil der Arbeit erledigt jedoch der Blitter – weshalb die BOBs ihren Namen zu Recht tragen.

Solche Datenverschiebungen könnte die CPU auch übernehmen – aber bei weitem nicht so schnell wie der Blitter. Eine CPU wie die M68000 kann Speicherbewegungen nämlich nur mit ganzen Gruppen von Bytes (zu 8 Bit) und Wörtern (zu 16 Bit) effizient und schnell durchführen (der Blitter ist allerdings auch bei solchen Gruppen schneller). Um ein unregelmäßig geformtes Objekt auf dem Bildschirm zu bewegen, muß man beliebige Häppchen von Wörtern und Bytes – unter Umständen sogar einzelne Bits – bewegen. Der Blitter kann Daten beliebiger Länge und an beliebigen Bit-Positionen im Speicher lesen und wieder hineinschreiben, ohne daß dies Geschwindigkeitseinbußen mit sich bringen würde. Die CPU würde bei Daten dieser Art sehr langsam werden.

Und weil der Blitter das Datenschaufeln so viel besser kann als die CPU, darf er dabei die CPU und alle anderen Chips auch vom Speicher »abschneiden«. (Dieses unhöfliche Verhalten des Blitters kann man allerdings auch abschalten.) Dadurch kann er dann Daten so schnell im Speicher hin- und herschieben wie die Speicherchips dazu überhaupt fähig sind. (Ansonsten müßte er sich ja den Speicher mit der CPU teilen, wie oben beschrieben, und würde nur die halbe Speichergeschwindigkeit nutzen können.) Das führt dazu, daß die CPU auf neue Daten aus dem Speicher warten muß, bis der Blitter fertig ist. Für Sie als Anwender oder Programmierer bedeutet das aber keine Geschwindigkeitseinbuße, wegen der man vom Gebrauch des Blitters absehen sollte. Immer, wenn der Blitter die CPU vom Speicher abschneidet, erledigt er Aufgaben, die die CPU in derselben Zeit sowieso nicht schaffen würde!

Der Blitter des Amiga 500 kann – im Gegensatz zu Blittern in anderen Computern – aber nicht nur Bereiche in einer Bitmap verschieben. Er kann bei diesem Verschiebevorgang auch Daten aus bis zu drei verschiedenen Quellen logisch miteinander kombinieren und die Ergebnisse dieser logischen Verknüpfung zurück in den Speicher schreiben. (Diese drei Quellen können verschiedene Bitmaps oder verschiedene Gebiete innerhalb einer Bitmap sein.) Erst das ermöglicht auch die Verschiebung völlig unregelmäßig geformter Flächen.

14.5.3 Der Copper

»Copper« ist eine Abkürzung für »Co-Processor«. Auch der Blitter ist natürlich ein Koprozessor, wie Sie gerade erfahren haben – aber für den Copper hatte man wohl keinen besseren Namen zur Hand. Der Copper ist allerdings ein ganz besonderer Koprozessor. Er ist nämlich an den Elektronen-Strahl gekoppelt, mit dem das Bild im Monitor gezeichnet wird und arbeitet zu diesem Zweck eng mit Denise zusammen.

Der Copper kontrolliert, während ein Bild auf den Bildschirm gezeichnet wird, permanent, wo sich der Elektronenstrahl gerade befindet (in welcher Zeile und Spalte des Bildes). Der Programmierer kann ihm befehlen, auf eine bestimmte Position zu warten, und kann bei Erreichen dieser Position Befehle an die anderen Chips schicken (indem er bestimmte Werte in die Register dieser Chips schreibt). Dadurch ergeben sich erstaunliche Möglichkeiten. Die Register des Chips Denise bestimmen ja zum Beispiel, welches Bild auf dem Bildschirm erscheint, welche Farben dafür zur Verfügung stehen und so weiter. Indem der Copper diese Register ändert, wenn der Elektronenstrahl auf halber Höhe des Bildes ist, kann er zum Beispiel die Farbpalette ändern. Dadurch können zum Beispiel in der oberen Hälfte des Bildes andere Farben als in der unteren verwendet werden.

Im Extremfall kann man den Copper aber nicht nur die Farbpalette, sondern alle Eigenschaften des Bildschirms – einschließlich des angezeigten Bildes – »mittendrin« ändern lassen. Das führt dann zu den Screens, die Sie ja schon im ersten Buchteil kennengelernt haben. Screens, eine Art virtuelle Bildschirme, erlauben es, die Ausgaben mehrerer Programme am Bildschirm zu betrachten, die völlig unterschiedliche Anforderungen an Bildschirmfarben und die Bildauflösung stellen. Als geschickter Programmierer kann man aber noch eine Menge anderer Tricks mit dem Copper anstellen, die Sie zweifellos schon in einigen Grafik-Programmen gesehen haben – ohne natürlich zu wissen, daß der Copper dafür zuständig ist.

14.6 Begriffsver- und -entwirrungen

Vielleicht ging es Ihnen in diesem Kapitel wie mir, als ich das erste Mal etwas über die Spezial-Chips des Amiga las und Sie leiden nun unter extremer Begriffsverwirrung. Ich hoffe, diese aber zum Abschluß dieses Kapitels noch etwas »entwirren« zu können. Vor allem möchte ich zwei Begriffe klären, die immer wieder verwendet werden, »Chip« und »Prozessor«.

Vielen dürfte im Moment nicht ganz klar sein, was denn nun ein Chip ist und wo die Koprozessoren des Amiga liegen. In der Berichterstattung über den Amiga in der Computer-Presse geht es meist auch wild durcheinander. Da ist von drei oder vier Custom-Chips die Rede: von Denise, Agnus (oder Agnes), von Paula, dem Blitter und dem Copper. Manchmal wurden der Blitter und der Copper als Chips bezeichnet und Denise und Agnus als Koprozessoren, und manchmal war es umgekehrt. In Wirklichkeit ist aber (hoffentlich) alles ganz einfach.

Paula, Denise und Agnus sind drei Chips im Innern des Amiga. Sie arbeiten aber so eng zusammen, daß sie eigentlich ein großes elektronisches Bauteil bilden und man sie deshalb am

besten auf einem Chip zusammenfassen müßte. Im Moment ist die Chip-Technologie aber noch nicht so weit, daß man einigermaßen preiswert einen so großen Chip bauen könnte. Aus rein praktischen Erwägungen sind die für den Amiga 500 benötigten grafischen und akustischen Funktionen deshalb auf drei getrennte Bausteine (Chips) verteilt worden.

Innerhalb der großen Funktionseinheit des Superchips, den Denise, Paula und Agnus zusammen bilden, existieren einzelne Untereinheiten, die sich um ganz spezielle Funktionen kümmern. Die Ansteuerung von externen Geräten, die Erzeugung eines Videobildes und die Manipulation einzelner Bits im Speicher sind zum Beispiel solche Funktionen, die von jeweils einer Untereinheit innerhalb des Superchips erledigt werden. Diese Untereinheiten nennen wir Prozessoren. Ein Prozessor ist eine *logische* Funktionseinheit, während ein Chip eine *physikalische*, greifbare Einheit ist (eben ein kleiner, schwarzer Klotz mit vielen Metallbeinchen).

Ein Prozessor kann komplett innerhalb eines der drei Chips liegen, kann sich aber auch aus Komponenten zusammensetzen, die auf mehrere Chips verteilt sind. Den Speicherzugriff für alle Aufgaben der verschiedenen Prozessoren erledigt zum Beispiel immer Agnus. Keine der anderen Funktionseinheiten in den anderen beiden Chips ist deshalb komplett ohne den Adreßgenerator in Agnus. Deshalb kann man eigentlich von keinem der Prozessoren, die oben besprochen wurden, sagen, daß er in einem Chip liegt.

Nur der Einfachheit halber spricht man oft davon, daß ein Prozessor in dem Chip liegt, der die »wesentlichen« Teile des Prozessors enthält. Denise wird deshalb nicht nur als Video-Chip, sondern manchmal auch als Video-Prozessor bezeichnet.

»Andersherum« können aber theoretisch durchaus mehrere Prozessoren innerhalb eines Chips liegen. Dies ist zum Beispiel beim Blitter und beim Copper der Fall, die sich im wesentlichen komplett in Agnus befinden. Wie Sie sehen, haben Chips und Prozessoren zwar viel miteinander zu tun, sind aber nicht identisch. Es sind Begriffe, die sich auf sehr unterschiedliche Eigenschaften beziehen. (Auch ein Mikroprozessor muß nicht auf einem Mikrochip liegen und ein Chip, den man als Mikroprozessor bezeichnet, kann mehrere verschiedene Prozessoren enthalten.)

Damit ist die Aufzählung der pingeligen Hardware-Details dann aber auch endgültig abgeschlossen. Ich hoffe, dem Leser in diesem Kapitel zumindest einen kleinen Eindruck von der Komplexität und Vielseitigkeit der Amiga-Hardware vermittelt zu haben, ohne dabei allzuviel Vorwissen vorausgesetzt zu haben.

14.7 Was bringt's ?

Nach dieser Flut von Details wird sich der Leser nun aber vielleicht fragen, was der ganze Aufwand denn nun wirklich bringt, beziehungsweise was der Amiga kann, was andere Computer, zum Beispiel der Atari ST, nicht können. Einen ungefähren Eindruck davon vermitteln vielleicht schon einige der Fotos in diesem Buch.

Gerade die Vielzahl der möglichen Farben – bei gleichzeitig hoher Bildschirmauflösung – läßt den Amiga in Bereiche vordringen, die bisher professionellen (und teuren) Systemen vorbehalten waren. Bei reinen Schwarzweiß-Bildern zum Beispiel wird schon fast Fernsehqualität erreicht und selbst Farbbilder wirken erstaunlich plastisch und natürlich – wenn auch etwas grobkörnig.

Hinzu kommt die Flexibilität der Amiga-Grafik-Architektur. Man kann einfache Grafik- oder Text-Bildschirme mit wenig Programm- und Speicheraufwand erstellen. Wahlweise kann man aber auch mit größerem Speicheraufwand – aber immer noch geringem Programmaufwand – sehr leistungsfähige Grafiksysteme erstellen. Und selbst bei diesen hat der Amiga oft noch genug Reserven frei, um mehrere Programme gleichzeitig bearbeiten zu können.

15 Die Programmierung des Amiga 500

In den vorangegangenen Kapiteln haben Sie ja schon einen ungefähren Eindruck davon bekommen, welche Möglichkeiten die Hardware des Amiga bietet. Falls der eine oder andere unter den Lesern dadurch Appetit bekommen hat, auch einmal selbst ein wenig mit diesen Möglichkeiten »herumzuspielen«, kann ihm vielleicht dieses und das nächste Kapitel helfen. Ab jetzt geht es nämlich um die Programmierung des Amiga 500. Hierzu werden Sie in diesem Kapitel zunächst einige allgemeine Erläuterungen zum Thema Programmierung finden. Im nächsten Kapitel wird dann stichwortartig beschrieben, was die Programmiersprache zu bieten hat, die jedem Amiga beiliegt: Amiga-BASIC.

Amiga-Programmierung spielt sich aber nie »im leeren Raum« ab. Alle Amiga-Programme nutzen intensiv die Systemsoftware, die sich auf der Workbench-Diskette befindet. Die »Kunst der Amiga-Programmierung« besteht unter anderem auch darin, diese »Bibliotheken« von Funktionen, die einem viel Arbeit abnehmen können, richtig zu nutzen.

15.1 Architektur der Amiga-Software

Im Zusammenhang mit der Software eines Computers von »Architektur« zu reden, mag vielleicht etwas hochtrabend klingen. In Wirklichkeit ist die Entstehung eines so komplexen Gebildes, wie es die Systemsoftware des Amiga darstellt, ein mindestens ebenso schwieriger und komplizierter Vorgang wie der Bau eines Hauses. Systemsoftware muß deshalb sehr sorgfältig geplant werden und verlangt ein reibungsloses Zusammenspielen vieler Beteiligter. Auch die Teile des entstehenden Gebäudes müssen nahtlos zusammenpassen und sich gegenseitig in genau berechneter Weise stützen.

Eine der größten Stärken des Amiga liegt in diesem reibungslosen Zusammenspiel der einzelnen Komponenten der Systemsoftware mit anderen Software-Komponenten und mit den Hardware-Komponenten. So arbeiteten zum Beispiel von Anfang an die beiden Teams, die die Hardware und die Software entwickelten, sehr eng zusammen und beeinflußten sich immer

wieder gegenseitig. Dies beschleunigte nicht gerade die Entwicklung, garantiert jetzt aber ein recht effizientes Softwaregebäude mit ideal aufeinander abgestimmten Bestandteilen.

Die »Kunst der Amiga-Programmierung« besteht zu einem großen Teil aus der geschickten Nutzung der Systemsoftware. Diese stellt eine »Software-Bibliothek« dar, in der Programme sich die Funktionen aussuchen können, die sie benötigen – und somit nicht noch einmal selbst realisieren brauchen. Sie als Programmierer brauchen also nicht »das Rad ein zweites Mal erfinden«.

15.1.1 Grundriß der Amiga-Systemsoftware

Damit Sie vielleicht den folgenden Erläuterungen ein wenig besser folgen können, möchte ich Ihnen zunächst einmal den Grundriß der Amiga-Systemsoftware vorstellen. Sie haben dann schon einmal einen ersten Überblick über dieses Gebäude. Die folgende Abbildung zeigt in schematischer Weise, wie die verschiedenen Komponenten dieser Systemsoftware aufeinander aufbauen.

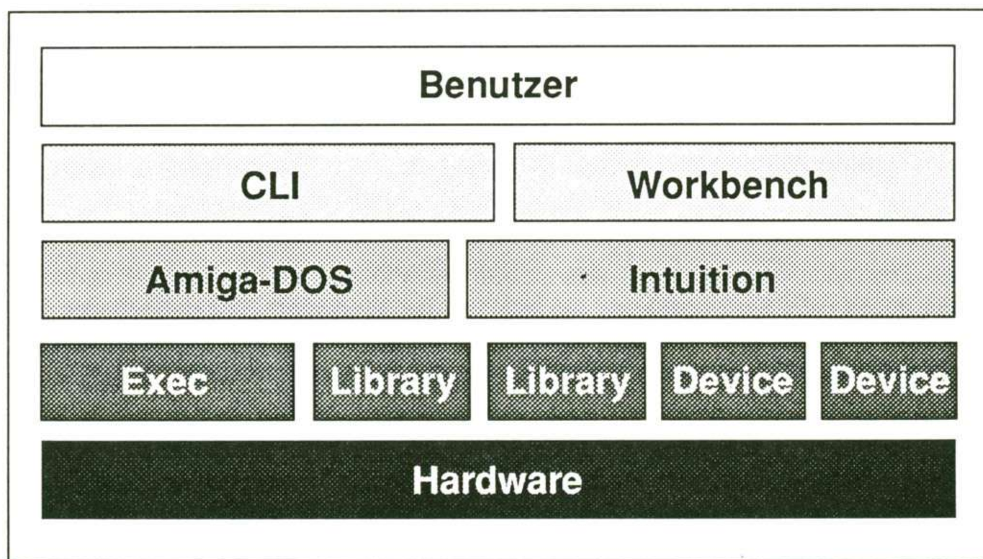


Bild 15.1: *Grundriß der Amiga-Systemsoftware*

»Ganz unten« in diesem Bild liegt die Hardware. Sie bildet praktisch das Fundament, auf dem die Software aufgebaut ist. Anders als bei simpleren Computern ist es bei Programmen für den Amiga 500 unüblich, daß Anwendungsprogramme direkt auf diese Hardware zurückgreifen. Das ist zwar möglich, denn die Hardware-Adressen und die Bedeutung der Register der Customchips sind gut dokumentiert, es ist aber viel komplizierter als zum Beispiel auf einem Commodore 64 und auch gefährlicher. Der Amiga ist nämlich eine Multitasking-Maschine

und wenn ein Programm nach seinem Willen die Custom-Chips manipuliert, versucht ein anderes vielleicht gleichzeitig etwas ganz anderes damit anzustellen, ohne vom ersten Programm etwas zu wissen. Das bedeutet Chaos (und abstürzende Programme).

Benutzen diese Programme jedoch die Amiga-Systemsoftware, kann nichts passieren. Die Systemsoftware kontrolliert den Zugriff auf die Hardware, spielt den Schiedsrichter zwischen den konkurrierenden Programmen und paßt auf, daß ein Programm nicht einem anderen in die Quere kommt. Das kann aber nur funktionieren, wenn diese Programme die Systemsoftware nicht umgehen und hinter ihrem Rücken Unsinn anstellen – wie zum Beispiel direkt auf die Hardware zurückzugreifen.

Direkt über der Hardware liegen *Exec* und eine Reihe von Bibliotheken und sogenannten »Geräte-Treibern«. Diese Schicht der Software hat direkten Zugriff auf die Hardware des Amiga 500. Die nächsthöhere Schicht, die aus *Intuition* und *Amiga-DOS* besteht, benutzt so die darunterliegende Schicht (Exec und die Bibliotheken), um die Hardware zu manipulieren.

Darüber kommen dann die Programme, mit denen Sie als Anwender direkt zu tun haben. Dazu gehören sowohl das CLI und die Workbench, wie natürlich auch alle anderen Anwendungsprogramme (Malprogramme, Textverarbeitungsprogramme und so weiter). Diese Programme benutzen vor allem Intuition, um mit Ihnen, dem Anwender, zu kommunizieren und AmigaDOS, um mit der Hardware des Amiga 500 umzugehen. Sie greifen zum Teil aber auch noch auf Exec und die Bibliotheken zurück. Über die Schicht von Workbench und CLI kommt dann keine weitere Software mehr, sondern direkt der menschliche Benutzer des Amiga 500 (also Sie selbst).

15.1.2 Exec

Die wichtigsten Komponenten der Hardware haben Sie ja in den vergangenen Kapiteln schon kennengelernt. Der wichtigste Bestandteil der Amiga-Systemsoftware ist Exec. Exec ist ein Paket von kleinen, sehr effizienten Routinen im ROM-Speicher des Amiga 500. Diese Routinen werden von allen anderen Komponenten der Software andauernd benötigt. Sie werden für eine Vielzahl von sehr elementaren Aufgaben bei der Verwaltung der verschiedenen »System-Ressourcen« des Amiga benötigt.

Listen

Exec bietet unter anderem Hilfsroutinen für die Verwaltung von Listen an. Solche Listen werden von fast jedem Programm und auch von anderen Teilen der Systemsoftware andauernd benötigt. In ihnen können beliebige Objekte, zum Beispiel Namen, Dateien, Programme, Fenster und so weiter gesammelt und verwaltet werden.

Da die Listenverwaltung sich innerhalb von Exec befindet, ist sie sehr schnell (in Maschinencode geschrieben) und steht jedem zur Verfügung, der sie benötigt. (Exec befindet sich ja immer im Speicher.) Andere Programme brauchen deshalb keine eigenen Pakete zur Listenverwaltung zu beinhalten und können dadurch kompakter werden.

Tasks

Eine besonders wichtige Sorte der »Objekte«, die Exec in Listen verwaltet, sind die Tasks, die verschiedenen Programme im Amiga, die scheinbar alle gleichzeitig laufen. Exec erzeugt und verwaltet solche Tasks, teilt ihnen Speicher und Rechenzeit auf der CPU zu und zerstört sie auch wieder, wenn sie ihre Aufgabe erfüllt haben.

Die Verteilung der CPU-Zeit (und anderer Ressourcen) auf die verschiedenen darum konkurrierenden Tasks ist eine der Hauptaufgaben von Exec. Auch hierbei zählt vor allem die Geschwindigkeit. Besonders der Wechsel von einem Task zu einem anderen muß sehr schnell vor sich gehen, da solche Wechsel jede Sekunde zigmal vorkommen können.

Nachrichten

Ebenfalls zur Task-Verwaltung gehört im weiteren Sinn auch der Nachrichten-Mechanismus von Exec. Er dient zu einer reibungslosen Kommunikation zwischen unterschiedlichen Tasks. Oft ist es nämlich nötig, daß verschiedene Tasks miteinander kommunizieren, um zum Beispiel Daten auszutauschen oder um sich mitzuteilen, wann wer mit welcher Aufgabe fertig ist. Dies ist nicht ganz so einfach, da kein Task vom anderen weiß, in welchem Zustand sich dieser gerade befindet. Ein Kommunikationsversuch zum falschen Zeitpunkt kann aber dazu führen, daß Daten verlorengehen oder den empfangenden Task bei einer wichtigen Aufgabe stören.

Der Nachrichten-Mechanismus von Exec umgeht diese Schwierigkeiten und sorgt dafür, daß Tasks nur dann Meldungen empfangen, wenn sie dazu bereit sind. Exec weiß natürlich als einziger immer ganz genau, ob ein Task gerade bereit ist, eine Nachricht zu empfangen oder nicht, da es die Tasks ja kontrolliert. Kann ein Task eine Nachricht nicht sofort empfangen, legt Exec sie erst einmal in eine Warteschlange (eine Liste; siehe oben), aus der sie der Task entnehmen kann, sobald er Zeit und Lust dazu hat. Keine Nachricht geht bei diesem Mechanismus verloren.

Speicherverwaltung

Exec kümmert sich zusätzlich auch noch um die Speicherverwaltung im Amiga. Es ist naheliegend, einem übergeordneten Paket wie Exec diese Aufgabe zu überlassen. Der Speicher ist schließlich eine sehr »globale« Ressource des Amiga, die von allen Programmen (Tasks) benötigt wird. Würden die verschiedenen Tasks gleichzeitig und unkontrolliert im Speicher herumfuhrwerken, wäre ein Chaos unvermeidlich.

Wann immer ein Programm mehr Speicherplatz benötigt (zum Beispiel, weil ein Benutzer eine neue Zeile in einem Textverarbeitungsprogramm eingegeben hat), kann es zusätzlichen Speicherplatz bei Exec anfordern. Exec sucht dann nach einem Stück im Speicher, das die passende Größe hat und noch frei ist, reicht dieses dem anfragenden Task und markiert es dann als belegt. Währenddessen werden alle anderen Programme angehalten, damit sie nicht zur selben Zeit denselben Speicher anfordern können.

»Brave« Programme geben einen so angeforderten Speicherblock natürlich auch sofort, wenn sie ihn nicht mehr benötigen, an Exec zurück. Dieser freigewordene Speicher kann dann anderen Programmen zugeteilt werden, die ihn vielleicht dringend brauchen.

15.1.3 Bibliotheken

Die wichtigste Aufgabe von Exec ist neben den Tasks aber wahrscheinlich die Verwaltung von Programmbibliotheken. Diese können in ähnlicher Weise von vielen Programmen gleichzeitig genutzt werden wie Exec selbst. Sie brauchen sich aber nicht in der Kickstart-Software zu befinden. Bibliotheken sind nichts anderes als besondere Dateien, die die Routinen einer Programmsammlung (Programm-Bibliothek) enthalten. Solche Dateien können jederzeit auf die Workbench-Diskette kopiert und danach von allen Programmen genutzt werden.

Exec sorgt dafür, daß Programme, die eine Bibliothek nutzen wollen, nicht zu wissen brauchen, wie diese Datei aussieht, wo sie liegt und ob sie bereits von einem anderen Programm benutzt wird oder neu in den Speicher geladen werden muß. Ein Programm braucht eine Bibliothek nur mit einem entsprechenden Exec-Aufruf und dem Namen der Bibliothek zu »öffnen« und kann danach die Routinen der Bibliothek benutzen, als wären es Bestandteile des Programms selbst.

15.1.4 Geräte

Bei der Beschreibung des CLIs haben Sie ja gesehen, daß die verschiedenen Ein- und Ausgabegeräte, die man am Amiga anschließen kann, behandelt werden können wie Dateien. Für diese Einheitlichkeit der Behandlung auch unterschiedlichster Geräte sorgen die Geräte-Bibliotheken (*devices*).

Für jedes angeschlossene Gerät richtet Exec nämlich zusammen mit dem sogenannten Treiber des Geräts einen separaten Task ein. Um Daten zu einem Gerät zu schicken oder welche von diesem zu holen, schickt Exec dem entsprechenden Task eine Nachricht, deren Format unabhängig von dem Gerät selbst ist. Diese Nachricht enthält nur die Art der Anforderung an das Gerät (also zum Beispiel, ob Lesen oder Schreiben eines Datenblocks gewünscht wird). Welche Disketten daraufhin gedreht oder welche Schalter umgelegt werden müssen, um diese Anforderung zu erfüllen, ist Exec egal. Der Geräte-Task ist alleine zuständig für die Beantwortung der Nachricht und somit für die exakte Ansteuerung der Hardware zu diesem Zweck.

Wenn ein neues Gerät an den Amiga angeschlossen wird, benötigt Exec nur einen passenden Treiber dazu. Dieser besteht einfach aus einer Datei, die der Geräte-Hersteller üblicherweise mitliefert und die in die Expansion-Schublade auf der Workbench-Diskette gelegt werden muß. Und schon können auch Programme Daten auf einem Gerät ausgeben, bei deren Entstehung es dieses Gerät vielleicht noch gar nicht gab. (In dieser Hinsicht sind Treiber durchaus ähnlich wie Bibliotheken zu verwenden.)

15.1.5 Die Grafik-Bibliothek

Eine dieser von Exec verwalteten Bibliotheken, die von fast allen Programmen genutzt wird, ist die Grafik-Bibliothek (engl. *Graphics Support Library*). Mit ihr hat direkt oder indirekt

jedes Programm etwas zu tun, das irgendwelche Ausgaben auf dem Bildschirm zeigen will – egal ob es sich um Texte oder Grafiken handelt.

Die Grafik-Bibliothek dient vor allem dazu, die Komplexität der Amiga-Grafik-Hardware vor dem Programm zu verbergen. Denn, obwohl die Fähigkeiten der Grafik-Hardware beachtlich sind, verlangen sie doch auch die Erledigung einiger Vorarbeiten und Haushaltungsaufgaben, bevor auch nur ein Punkt oder kurzer Strich auf dem Bildschirm erscheint.

Durch die Routinen der Grafik-Bibliothek sieht der Bildschirm auch für ein Programm so simpel aus wie eine Malfläche, über die ein Pinsel oder Stift bewegt wird. Nur die Größe dieser Fläche und die Anzahl der Farben in der Palette müssen vor der ersten Zeichenoperation festgelegt werden. Um danach zum Beispiel einen Strich zu zeichnen, wählt man auch als Programmierer zunächst eine Farbe aus. Der Zeichenstift kann dann durch Aufruf der entsprechenden Routinen zu jedem beliebigen Punkt der Fläche bewegt werden. Bei jeder solchen Bewegung des Stiftes kann man wählen, ob der Stift über der Zeichenfläche »schwebt« und somit nur bewegt wird, oder ob er »abgesenkt« wird und Spuren – also eine Linie – hinterläßt.

Andere Grafik-Routinen sind fähig, Rechtecke zu zeichnen und beliebige Flächen, die durch Punkte einer anderen Farbe begrenzt sein müssen, mit einer beliebigen neuen Farbe zu füllen. Letzteres funktioniert wie der Farbeimer in GraphiCraft oder DeluxePaint (und ist auch genauso empfindlich gegenüber »Lecks« in der Umrandung).

Und schließlich ist die Grafik-Bibliothek auch noch für das »Malen« von Text auf dem Bildschirm zuständig. Dies ist bei bitmapped Grafik ja nicht so einfach wie bei Computern, die einen reinen Text-Modus besitzen. Erschwerend kommt noch hinzu, daß der Amiga viele verschiedene Schriftarten, Schriftstile und -größen kennt. Bei Verwendung der entsprechenden Routinen aus der Grafik-Bibliothek ist ein Text aber so einfach wie ein Strich zu zeichnen. Zunächst wird die gewünschte Schriftart mit ihrem Namen ausgewählt und die Größe der Buchstaben sowie der Textstil (fett, unterstrichen, kursiv oder eine Kombination davon) festgelegt. Dann wird der Stift zu der Position bewegt, wo der Text beginnen soll und eine Routine der Grafik-Bibliothek aufgerufen, der der Text übergeben wird. Diese zeichnet ihn dann selbständig auf den Bildschirm, wobei für die Buchstaben die aktuelle Farbe des Zeichenstiftes verwendet wird.

15.1.6 Intuition

Auch Intuition ist eine Bibliothek – eine sehr wichtige allerdings. Was Intuition so alles leistet, ist ja (zum Teil) bereits im Kapitel 6 aufgelistet worden. Intuition ist der Teil der Systemsoftware, mit dem ein Programm wahrscheinlich am meisten kommuniziert. Exec und die Grafik-Bibliothek werden zwar andauernd benötigt, ein Programm merkt davon aber im allgemeinen recht wenig und hat auch nicht direkt mit Exec zu tun. Vieles hingegen »läuft« über Intuition-Routinen.

Intuition enthält zum Beispiel Routinen zum Öffnen und Schließen von Screens, Fenstern und Requestern. Auch alle Gadgets und Menüs werden von Intuition gezeichnet und bearbeitet. Die Routinen in Intuition achten auch darauf, daß alle Ausgaben eines Programms in dem ihm

zugeteilten Fenster oder Screen bleiben. Eine lange Linie, die von einem Programm gezeichnet wird, stoppt Intuition zum Beispiel automatisch am Fensterrand. Durch Intuition wird so die (pseudo-) parallele Arbeit mit mehreren Programmen in mehreren Fenstern erst möglich. Intuition sorgt dafür, daß sich die Ein- und Ausgaben gleichzeitig laufender Programme in säuberlich begrenzten Gebieten auf dem Bildschirm abspielen und sich deshalb nicht gegenseitig stören.

Intuition nimmt einem Programm zusätzlich auch noch sehr viel Arbeit ab bei der Kommunikation mit dem Anwender des Programms. Alle Aktionen des Anwenders – vor allem Mausklicks und Tastendrücke – laufen nämlich zunächst einmal durch Intuition, wo sie einer Vorverarbeitung unterzogen werden. Mausklicks in ein Gadget, die Titelleiste eines Screens oder Fensters und so weiter, bearbeitet Intuition zunächst einmal selbst. Vom Bewegen und Verkleinern eines Fensters, vom Herunterklappen eines Menüs merkt ein Programm recht wenig. Und dies ist auch gut so, weil die damit verbundenen Operationen recht komplex sind (obwohl alles so mühelos aussieht) und das Programm nur von der Erledigung seiner eigentlichen Aufgaben abhalten würden.

Erst, wenn eine Reaktion des Programms nötig wird – zum Beispiel wenn der Anwender einen Menübefehl ausgewählt oder eine Taste gedrückt hat – wird die entsprechende Aktion in Form einer Exec-Nachricht »verpackt« und an den Programm-Task des aktiven Fensters geschickt. Diese Nachricht enthält zum Beispiel die Nummer des ausgewählten Menüs und des Befehls darin oder den Buchstaben, der einer soeben gedrückten Taste entspricht. Was ein Programm mit dieser Nachricht macht, ist dann seine Sache.

15.1.7 Amiga-DOS

Ein weiterer wichtiger Bestandteil der Amiga-Systemsoftware ist Amiga-DOS. Die Bezeichnung »Amiga-DOS« ist eine Abkürzung für *Amiga Disk Operating System*. Das sagt auch schon aus, wofür Amiga-DOS vor allem zuständig ist. Zusammen mit den oben beschriebenen Gerätetreibern (*devices*) sorgt Amiga-DOS dafür, daß sich die Programme beim Zugriff auf die Amiga-Hardware (zum Beispiel die Diskettenlaufwerke) nicht um Details kümmern brauchen. Für Programme sehen Dateien deshalb aus wie ein kontinuierlicher Strom von Daten. Nur Amiga-DOS braucht zu wissen, wie diese Daten auf der Diskette verteilt sind (sie können wild durcheinanderliegen).

Amiga-DOS kann neue Dateien und Directories auf Disketten und Festplatten anlegen, sie wiederfinden, wenn man den (Pfad-) Namen kennt, sie wieder löschen und umbenennen. Und schließlich sorgt Amiga-DOS – wieder zusammen mit den Gerätetreibern – dafür, daß fast jedes Ein-/Ausgabe-Gerät wie eine Datei behandelt werden kann.

15.2 Grundlagen der Programmierung

Zur Programmierung des Amiga benötigt man, wie für jeden anderen Computer auch, eine Programmiersprache. In Wirklichkeit versteht der Amiga 500 nur eine (Programmier-) Sprache, den sogenannten Maschinencode. Dies ist eine Folge von Bits, die zu »Wörtern« von

jeweils 16 Bit zusammengefaßt werden. Da ein Mensch normalerweise nicht in Bit-Folgen denkt und redet, spielt sich die Kommunikation mit dem Computer zwangsläufig etwas schleppend ab, wenn nicht zusätzliche Hilfsmittel eingesetzt werden.

15.2.1 Sprachen

Ein solches Hilfsmittel stellen die sogenannten »höheren Programmiersprachen« dar – höher deshalb, weil man sich nicht mehr auf das »niedere« Niveau der Maschine hinabbegeben muß. Die wesentliche Idee bei allen höheren Programmiersprachen ist es, ein Programm zunächst einmal in einer einigermaßen lesbaren Sprache zu formulieren, die aus »richtigen« Wörtern besteht, welche aus »richtigen« Buchstaben gebildet werden. Wie nahe die so entstehende Sprache der menschlichen Sprache kommt, ist verschieden. Es gibt sehr primitive Programmiersprachen, die wohl nur von Fachleuten verstanden werden und sehr »hohe« Programmiersprachen, in denen sich Programme fast wie Gebrauchsanweisungen aus dem vielzitierten »richtigen Leben« lesen.

Falls Sie bisher noch keine Programmiersprache kennen, lassen Sie sich also nicht vom Getue der Programmierer abschrecken, eine zu erlernen. Alle Programmiersprachen sind viel simpler und leichter zu erlernen als auch die einfachste menschliche Sprache.

Was alle Programmiersprachen bei all ihren Unterschieden gemeinsam haben, ist, daß immer ein Verfahren definiert ist, wie man von einer Anweisung in der höheren Programmiersprache zur äquivalenten (gleichbedeutenden) Bitfolge im Maschinencode kommt. Äquivalent heißt in diesem Fall, daß die Bitfolge das tut, was der Programmierer gemeint hat, als er das Programm in der höheren Sprache aufgeschrieben hat. (Eines der Hauptprobleme bei der Programmierung ist, daß sich Programmierer und Computer meist nicht so ganz darüber einig werden können, was der Programmierer denn nun eigentlich »gemeint« hat!)

Das Verfahren, das ein Programm in Maschinencode umwandelt, ist relativ (!) simpel und geradlinig. Es ist zumindest so simpel, daß man es wiederum einem Computer-Programm überlassen kann. Es gibt zwei große Gruppen solcher Programme, die zwei grundsätzliche Strategien repräsentieren, mit denen man diesen Übersetzungsprozeß angehen kann: Interpreter und Compiler.

Eine Ursache für viel Verwirrung bei Programmieranfängern ist oft, daß diese Programme, die andere Programme in Maschinencode umsetzen, ebenfalls Programmiersprache genannt werden. Dies ist eigentlich falsch! Genau wie ein Dolmetscher für Französisch nicht die Sprache Französisch ist, ist auch ein Interpreter für BASIC nicht die Sprache BASIC. Behalten Sie das immer im Auge, wenn Ihnen jemand das nächste Mal erzählen will, diese oder jene Computer-Sprache sei gut oder schlecht. Meist will er damit in Wirklichkeit sagen, daß der Compiler oder Interpreter gut oder schlecht ist.

15.2.2 Interpreter

Ein Interpreter ist die einfachste Möglichkeit, eine Umsetzung von höherer Programmiersprache zum Maschinencode zu realisieren. Er enthält üblicherweise für jede Anweisung (jedes Wort) der höheren Programmiersprache ein kleines Stück Maschinencode,

das dieser Anweisung entspricht (ihr Äquivalent ist). Um ein Programm auszuführen, liest ein Interpreter es Zeile für Zeile, sucht die den Anweisungen entsprechenden Maschinencode-Stückchen heraus und übergibt diese dem Computer zur Ausführung. Diesen Vorgang nennt man dann sinnigerweise auch »Interpretieren« eines Programms.

Wenn man einen Vergleich mit menschlichen Sprachen ziehen würde, könnte man sagen, daß ein Interpreter einem Dolmetscher entspricht, der ein fremdsprachiges Buch vorliest und dabei simultan übersetzt.

15.2.3 Compiler

Ein Compiler hingegen nimmt den Programmtext (in der höheren Programmiersprache) und erzeugt daraus ein äquivalentes Programm in Maschinencode. Der gesamte Übersetzungs-Vorgang heißt »Kompilieren« oder auch einfach nur »Übersetzen«. Dabei werden raffinierte Techniken angewandt, um den Maschinencode möglichst klein und schnell zu machen. Im allgemeinen produzieren diese Techniken nicht aus einer Programmanweisung immer ein und dasselbe Stückchen Maschinencode, sondern verwenden Wissen über die vorangehenden und nachfolgenden Anweisungen, um den erzeugten Code zu optimieren.

Wollte man wieder den Vergleich mit dem fremdsprachigen Buch bemühen, könnte man sagen, daß ein Compiler einem Dolmetscher entspricht, der eine schriftliche Übersetzung des Buches anfertigt, die man danach selbst lesen kann, ohne dazu den Dolmetscher noch zu benötigen.

15.2.4 Vergleich von Compiler und Interpreter

Die Vorteile des Compiler-Prinzips liegen auf der Hand. Der erzeugte Maschinencode ist selbständig und benötigt kein zusätzliches Programm (den Interpreter) zur Ausführung. Zudem sind kompilierte Programme üblicherweise fast immer schneller, da kein Interpreter mühselig Zeile für Zeile den Programmtext durchlaufen, analysieren und interpretieren muß.

Compiler haben aber auch gravierende Nachteile, weshalb sie sich bisher trotz ihrer Vorteile noch nicht generell für alle Sprachen durchgesetzt haben. Der Vorgang des Kompilierens eines Programms dauert zum Beispiel oft sehr lange. Das dabei entstehende Programm ist zwar schnell, dafür muß man zunächst aber erst einmal eine Wartezeit in Kauf nehmen. Dies ist besonders während der Programmentwicklung sehr lästig. Nach jeder kleinen Änderung muß das Programm neu übersetzt werden und man muß oft mehrere Minuten warten, bis die Änderungen am laufenden Programm ausgetestet werden können. Bei Verwendung eines Interpreters ist ein Programm auch nach einer Änderung sofort wieder lauffähig.

Compiler sind zudem meist schwieriger zu bedienen als Interpreter. Während es bei Interpretern in der Regel ausreicht, einen Start-Befehl (zum Beispiel RUN) zu geben, um ein Programm mit der Arbeit beginnen zu lassen, erfordern Compiler mehrere Arbeitsschritte bis zu diesem Punkt (bei denen man natürlich auch jedesmal Fehler machen kann). Interpreter sind deshalb auch heute noch unschlagbar, wenn es darum geht, eine Programmiersprache zu erlernen oder ein Programm wirklich interaktiv zu entwickeln und auszutesten.

15.3 Programmiersprachen für den Amiga 500

Für den Amiga 1000 (das Vorgängermodell des Amiga 500) gab es schon kurze Zeit nach der Markteinführung eine Vielzahl von Compilern und Interpretern – deren Zahl ist im Laufe der Zeit natürlich noch angewachsen. Für den Amiga 500 sind deshalb alle weitverbreiteten Computersprachen zu haben. Kein Programmierer wird also durch fehlende Werkzeuge davon abgehalten, sich mit den phantastischen Möglichkeiten des Amiga näher zu beschäftigen.

15.3.1 Assembler und Maschinencode

Die Programmiersprache, die dem Maschinencode noch am nächsten kommt, ist die Assembler-Sprache. Bei einem Assembler entspricht jede Zeile des Programms ziemlich genau einer Instruktion (einem Wort) im Maschinencode. Der Compiler (in diesem Fall ebenfalls »Assembler« genannt) hat es also mit dieser Sprache recht leicht.

Mit einem guten Assembler kann man wirklich alle Möglichkeiten nutzen, die ein Computer zu bieten hat, muß sich dabei aber auf ein recht niedriges Niveau der Programmierung hinabbegeben. Selbst einfache Dinge – wie das Ausgeben eines kurzen Textes auf dem Bildschirm – erfordern in Assembler-Sprache oft nicht-triviale, größere Programme.

Ein weiteres Problem der Assembler-Sprache ist, daß es sehr leicht ist, beliebig viele logische Fehler in ein Programm einzubauen; der Assembler merkt nichts davon. Höhere Programmiersprachen schützen in stärkerem Maße vor den eigenen Fehlern und nehmen einem damit die lästige Arbeit der Fehlersuche zumindest zum Teil ab.

Nur wenige Programme werden heute noch in Assembler-Sprache entwickelt, da es inzwischen andere Programmiersprachen gibt, mit denen die meisten Programmieraufgaben genauso gut erledigt werden können wie in Assembler-Sprache. Nur wenn Geschwindigkeit und Kompaktheit eine extrem große Rolle spielen (oder der Programmierer einfach Spaß am »Bit-Popeln« hat) greift man heute noch auf die Assembler-Sprache zurück.

Ein Assembler gehört zum Standard-Lieferumfang des Entwickler-Paketes, das Software-Entwickler bei Commodore erwerben können. Beim Amiga für »normale Sterbliche« liegt kein Assembler bei. Er kann aber jederzeit mit den dazugehörigen Unterlagen auch separat erworben werden. Bei diesem Assembler handelt es sich ansonsten um einen recht komfortablen Vertreter seiner Gattung (für Fachleute: es ist ein vollausgebauter Makro-Assembler). Auch andere Firmen bieten inzwischen eine ganze Reihe verschiedener Assembler (zum Teil mit »Monitor« oder »Debugger«) für den Amiga 500 an.

15.3.2 BASIC

BASIC dürfte heute die am weitesten verbreitete Computersprache auf der ganzen Welt sein. Sie gehört standardmäßig zum Lieferumfang vieler Computer, so auch des Amiga, oder wird sogar gleich in die Computer eingebaut (in Form eines ROMs). Über Amiga-BASIC erfahren Sie mehr im folgenden Kapitel, das sich der Sprache und dem zugehörigen Interpreter ausführlich widmet. Diese Sprache liegt dem Amiga 500 beim Kauf gleich bei. Wem sie nicht gefällt,

der bekommt inzwischen einige Alternativen dazu auf dem Markt angeboten, die verschiedene Vorteile (und Nachteile) gegenüber Amiga-BASIC haben.

15.3.3 C

Eine Sprache, die sich in den letzten Jahren gerade auf Mikrocomputern wachsender Beliebtheit erfreut, ist auch C. Sie stellt einen – nach Meinung vieler Fachleute sehr gelungenen – Kompromiß zwischen Maschinennähe (à la Assembler) und Programmierkomfort dar. C wird fast nur im Zusammenhang mit Compilern verwendet und ist dementsprechend auch nicht gerade eine Sprache für Anfänger.

Ein zusätzlicher Nachteil von C aus der Sicht eines Anfängers ist, daß es, ähnlich wie beim Assembler, wenig Schutz vor den eigenen Fehlern gibt. Man kann viele Fehler in ein Programm einbauen, die so «dumm« sind, daß sie ein derart aufwendiges Programm (wie es ein guter C-Compiler darstellt) eigentlich finden könnte – nur: er sucht gar nicht danach!

Dafür erzeugen C-Compiler aber meist sehr »guten« Maschinencode, der recht nahe an die Geschwindigkeit und die Kompaktheit herankommt, die man erreichen könnte, wenn man dasselbe Programm mit einem Assembler erstellen würde – was ungleich schwieriger und langwieriger wäre. Zudem sind die C-Compiler vieler Hersteller so raffiniert konstruiert, daß man sie sehr leicht an einen neuen Computer anpassen kann. Dies gilt selbst dann, wenn dieser einen ganz anderen Maschinencode versteht. Es gibt deshalb für neue Computer meist sehr schnell C-Compiler, während Compiler und Interpreter für alle anderen Sprachen oft lange auf sich warten lassen.

Genau das war auch beim Amiga der Fall. Für den Amiga 500 (mit externem Laufwerk) gibt es inzwischen mindestens zwei C-Compiler, die beide sehr gut sind. Einen dieser Compiler gab es schon, bevor der Amiga 1000 überhaupt auf dem Markt war.

15.3.4 Pascal und Modula

Ebenfalls für die Ausbildung – in diesem Fall aber die Ausbildung von Informatikern an der ETH Zürich – wurde die Programmiersprache Pascal konzipiert. Pascal ist nicht ganz so leicht zu erlernen wie zum Beispiel BASIC, zwingt den Programmierer aber mehr oder weniger zu strukturierten und leserlichen Programmen. In Pascal sind zudem diverse Konzepte »eingebaut«, durch die viele Fehler, die die Compiler anderer Sprachen einfach »durchgehen lassen«, schon bei der Übersetzung des Programms erkannt werden können.

Jedem, der nicht nur »schnell mal eben« eine Programmiersprache, sondern das »Programmieren an sich«, in fundierter Weise lernen möchte, kann Pascal nur wärmstens empfohlen werden. Wer mit Pascal das Programmieren gelernt hat, hat meist auch nur wenig Schwierigkeiten, eine weitere Sprache zu lernen – was man vom umgekehrten Weg leider nicht behaupten kann.

Auch für den Amiga gibt es einen Pascal-Compiler (MCC-Pascal). Dieser entspricht hundertprozentig dem internationalen Sprach-Standard, wie ihn die ISO (International Standards Organisation) definiert hat und hat den zusätzlichen Vorteil, Programmstücke in

Pascal, C und Assembler-Sprache innerhalb eines Programms kombinieren zu können. Er bietet aber leider nur sehr umständlichen Zugriff auf die speziellen Fähigkeiten des Amiga. Amiga-Besitzer warten deshalb immer noch sehnlichst auf das legendäre Turbo-Pascal, das es hoffentlich ab Anfang nächsten Jahres (1988) auch für den Amiga 500 geben wird.

Pascal war aber nur der Anfang einer ganzen Sprachfamilie. Nach einer gewissen Zeit der Verwendung von Pascal stellt sich nämlich heraus, daß es bei allen Vorteilen dieser Sprache ihr doch an einigen wichtigen Fähigkeiten für bestimmte Gruppen von Programmen mangelte. So wird zum Beispiel die Entwicklung großer Programmprojekte durch ganze Gruppen von Programmierern kaum unterstützt und es gibt kaum Möglichkeiten, auf die Hardware des verwendeten Computers zurückzugreifen.

Die Entwickler von Pascal nahmen dies zum Anlaß, eine neue Sprache zu entwerfen, die zwar nicht als Ersatz, aber auf vielen Gebieten als Ergänzung zu Pascal gesehen werden kann: Modula (beziehungsweise Modula-2). Sie beseitigt weitgehend alle als schwerwiegend empfundenen Mängel von Pascal und hat dabei aber genügend Ähnlichkeiten mit Pascal, um vom Programmierer nur minimale Umstellungen zu verlangen. Auch für den Amiga 500 konnten interessierte Programmierer zum Zeitpunkt, als dies Buch entstand, schon zwei verschiedene Modula-2-Systeme erwerben. Auch hier – wie bei allen anderen Sprachen – steht man also vor der Qual der Wahl.

16 Amiga-BASIC

Die Programmiersprache des Amiga 500 heißt Amiga-BASIC. Das zugehörige Werkzeug befindet sich auf der Extras-Diskette (sie trägt auf der Workbench den Namen *ExtrasD*), die jedem Amiga 500 beim Kauf beiliegt und gibt so jedem Besitzer des Amiga die Gelegenheit, eigene Programme zu entwickeln oder auch nur ein wenig mit den Möglichkeiten der Computerprogrammierung herumzuspielen. Sie werden Amiga-BASIC in diesem Kapitel etwas näher kennenlernen.

Der Leser möge mir aber verzeihen, wenn er hier keine ausführliche Einführung in die Computer-Programmierung findet, ja noch nicht einmal eine Anleitung zur Programmierung des Amiga 500. Programmierung ist eine Sache, die man nicht in einem Buchkapitel abhandeln kann. Und Amiga-BASIC ist eine so umfangreiche und vielseitige Sprache, daß selbst ein Buch vom Umfang des hier vorliegenden sich wahrscheinlich noch recht knapp fassen müßte, wenn es Amiga-BASIC vollständig beschreiben wollte.

Die folgenden Abschnitte geben deshalb nur einen kurzen Überblick über die Möglichkeiten der Sprache BASIC und des speziellen Dialekts Amiga-BASIC. Sie werden so unter anderem erfahren, welche Möglichkeiten des Amiga Sie in eigenen Programmen nutzen können (fast alle!) und wie Amiga-BASIC gegenüber anderen Dialekten der Sprache BASIC auf anderen Computern einzuordnen ist.

Die letzten Abschnitte dieses Kapitels stellen schließlich einige Beispielprogramme vor, mit denen Sie versuchen können, mit Amiga-BASIC »etwas warm zu werden«. Diese Programme leisten alle nichts Weltbewegendes, sondern dienen wirklich nur zu Demonstrationszwecken. Falls Sie wirklich eigene Programme für den Amiga entwickeln wollen, so erhalten Sie Informationen dazu zusammen mit dem Amiga 500 selbst im relativ ausführlichen Handbuch für Amiga-BASIC. Falls Sie noch keinerlei Erfahrung mit der Programmierung von Computern haben, wird dieses Handbuch allerdings nicht ausreichen. Für die Einführung in die Sprache BASIC existieren aber bereits einige sehr gute Lehrbücher, die Ihnen in diesem Fall helfen sollten. Speziell zu Amiga-BASIC gibt es das Buch von David A. Lien

»Programmier-Praxis mit MS-BASIC« (tewi-Verlag, München, 1986, Best.-Nr. 80369). Viele Volkshochschulen bieten zudem inzwischen ebenfalls BASIC-Kurse an. Diese sind zwar nicht speziell für den Amiga ausgelegt, wenn Sie aber erst einmal ein BASIC kennengelernt haben, werden Ihnen die anderen Dialekte keine allzu großen Schwierigkeiten mehr bereiten. Es könnte sogar sinnvoll sein, zunächst eine ganz andere Programmiersprache, wie zum Beispiel Pascal, zu erlernen. Die Unterschiede zwischen den verschiedenen Computer-Sprachen zu bewältigen, ist bei weitem nicht so schwer wie ein Wechsel von Deutsch nach Französisch.

16.1 BASIC

BASIC ist mehr oder weniger die Computersprache, die für die meisten Leute untrennbar mit dem Mikrocomputer verbunden ist. Sie hatte aber schon eine lange Geschichte hinter sich, bevor die erste Zeile eines BASIC-Programms auf einem Mikrocomputer ausgeführt wurde.

Das Wort BASIC ist eine Abkürzung für ***B**eginners **A**ll **P**urpose **S**ymbolic **I**nstruction **C**ode* (Allzweck-Instructionscode für Anfänger) und deutet damit schon auf den Zweck hin, zu dem die Sprache ursprünglich am Dartmouth-College entwickelt wurde: der Ausbildung von Anfängern in der Computer-Programmierung (in einem *Instructionscode*, wie man damals noch sagte). Auch heute noch wird BASIC vielfach für diese Zwecke eingesetzt. Nach wie vor ist einer der Hauptvorteile dieser Sprache nämlich die schnelle Erlernbarkeit. Das führt dazu, daß jeder Anfänger relativ schnell eigene Programme schreiben und so seine ersten Erfolgserlebnisse am Computer haben kann.

Die ersten BASIC-Realisierungen für Mikrocomputer waren aber (aus Gründen des damals noch sehr knappen Speicherplatzes) notwendigerweise sehr beschränkt. BASIC-Dialekte auf Mikrocomputern waren lange Zeit recht primitive Sprachen, die wirklich nur die allernötigsten Hilfsmittel der Programmierung zur Verfügung stellten. Das führte in der Folgezeit dazu, daß BASIC in Kreisen der Computer-Wissenschaft (in Deutschland heißt sie Informatik) einen sehr schlechten Ruf bekam. Mit diesen primitiven BASIC-Formen war es nämlich sehr leicht, Programme zu schreiben, die so unübersichtlich waren, daß sie nach kurzer Zeit niemand mehr verstehen und vor allem ändern konnte. Gerade das Verstehen und Ändern alter Programme ist aber eine der Hauptaufgaben vieler Programmierer und kostet die Programm-Hersteller auf Dauer mehr als das Erstellen einer ersten Programmversion.

Einer der Vorwürfe, die man BASIC oft machte, war zum Beispiel, daß die Sprache zu sogenanntem »Spaghetti-Code« verführt. Viele Programme in den frühen BASIC-Versionen waren nämlich lange Ketten von unstrukturierten Programmzeilen, zwischen denen das Programm bei der späteren Ausführung hin- und herhüpfen mußte. Wenn man diese Programmsprünge auf Papier mit einem Bleistift nachzuvollziehen versuchte, ergab dies ein wildes Durcheinander von Linien – eben den berühmten Spaghetti-Code.

Zudem kannten frühe BASIC-Versionen keine Namen für Teilprogramme (Subroutinen oder Prozeduren), sondern nur Nummern. Stand in einem anderen Programmteil nun der Aufruf einer solchen Prozedur, mußte man erst die Bedeutung dieser Nummer nachschlagen, wenn

man wissen wollte, was das Programm in diesem Teil tun sollte. (Oft war diese Bedeutung sogar überhaupt nirgends dokumentiert und nur der Programmierer allein wußte, was das entsprechende Unterprogramm tun sollte.) In anderen Programmiersprachen war es zu dieser Zeit längst üblich, Unterprogrammen sinnvolle Namen zu geben und sie auch mit diesen Namen aufzurufen. Statt *GOSUB 5390* in BASIC konnte man schreiben *Zeige_Daten_Von(Müller)*.

Im Laufe der Zeit wurde BASIC aber auch auf Mikrocomputern immer »erwachsener», was nicht zuletzt damit zusammenhing, daß der Speicher immer billiger wurde, und die BASIC-Programme größer wurden und so auch mehr Möglichkeiten enthalten konnten. Das erste BASIC auf einem Mikrocomputer war zum Beispiel gerade 4 Kbyte groß; Amiga-BASIC hingegen ist immerhin schon 80 Kbyte groß und es ist nicht einmal das größte BASIC auf Mikrocomputern.

Viele heutige BASIC-Dialekte (wie zum Beispiel Amiga-BASIC) enthalten alle wesentlichen Merkmale einer modernen Programmiersprache. Sie erlauben mühelos die sogenannte strukturierte Programmierung, mit deren Hilfe Programme so übersichtlich geschrieben werden können, daß man sie auch noch Jahre, nachdem sie geschrieben wurden, verstehen und ändern kann. Unterprogramme und Programmvariablen können sinnvolle Namen erhalten und eine Vielzahl wichtiger mathematischer und anderer Funktionen sind fest eingebaut und brauchen nicht erst mühselig von jedem Programmierer neu geschrieben zu werden.

BASIC hat im Laufe der Jahre viele gute und erprobte Fähigkeiten anderer Programmiersprachen aufgenommen und läßt auch hinsichtlich des Komforts bei der Programmentwicklung und Fehlersuche heute nicht mehr viele Wünsche offen. Im nächsten Abschnitt wollen wir uns den Dialekt »Amiga-BASIC» einmal genauer betrachten, um zu sehen, wie er im Vergleich abschneidet.

16.2 Der BASIC-Dialekt »Amiga-BASIC«

Das BASIC des Amiga stammt von der Firma Microsoft®. Dieses Softwarehaus ist bezüglich der Sprache BASIC praktisch der »Hauslieferant» für fast alle Hersteller von Mikrocomputern. Das erste BASIC für einen der ersten Mikrocomputer stammte von Microsoft, und bis heute gibt es kaum einen Mikrocomputer, für den es kein Microsoft-BASIC (MS-BASIC) gibt. Es ist meist die erste Programmiersprache, die für jeden neuen Computer auf den Markt kommt.

16.2.1 Microsoft-BASIC

Diese weite Verbreitung von MS-BASIC ist ein enormer Vorteil für den Programmierer, der diese Sprache benutzt. Alle BASIC-Versionen der Firma Microsoft sind sich nämlich (jeweils in der neuesten Version) weitgehend ähnlich. Wenn man Microsoft-BASIC auf einem Computer kennt und dann auf einen anderen überwechselt, gibt es deshalb sehr wenig Umstellungsschwierigkeiten. (Dies betrifft natürlich nicht die speziellen Fähigkeiten der verschiedenen Computer, die natürlich in den verschiedenen BASICs auch unterschiedlich unterstützt werden müssen. Viele andere Computer haben nun einmal keine Maus, Fenster und

hochauflösende Farbgrafik wie der Amiga.) Viele Programme können deshalb mit sehr wenigen Änderungen direkt von einem Computer zum anderen übernommen werden, so daß man besonders bei neuen Computern gleich zu Anfang schon etwas Software zur Verfügung hat.

Auch wenn Sie ein gutes Lehrbuch für Amiga-BASIC suchen, hilft Ihnen die weite Verbreitung von MS-BASIC. Es dauert nämlich naturgemäß einige Zeit, bis gute Programmierbücher für einen neuen Computer auf den Markt kommen. Falls Sie aber ein gutes BASIC-Buch für den IBM-PC® (PC-BASIC® oder MS-BASIC®) oder noch besser den Apple® Macintosh® finden, zögern Sie nicht, zuzugreifen! Alle »wesentlichen« Eigenschaften des BASIC für den IBM-PC und den Amiga sind gleich. Die Ähnlichkeiten bezüglich der Verwaltung von Menüs, Maus und Fenstern zwischen Amiga-BASIC und MS-BASIC für den Apple Macintosh sind sogar noch größer. Ansonsten kann ich an dieser Stelle nur noch einmal »Programmier-Praxis mit MS-BASIC« von David A. Lien (tewi-Verlag, München, 1986, Best.-Nr. 80369) empfehlen.

16.2.2 Moderne Features von Microsoft-BASIC

Einige der wichtigsten Eigenschaften des MS-BASIC für den Amiga sollen nun stichwortartig aufgelistet werden. Es gibt sie fast alle in jeder neueren Version von MS-BASIC und sie sind deshalb nicht Amiga-spezifisch. Es sind aber Möglichkeiten, die durchaus nicht in jedem BASIC zu finden sind, und die deutliche Fortschritte gegenüber den primitiven frühen BASIC-Versionen (siehe oben) darstellen.

- Optionale Zeilennummern: Alle Zeilen eines Programms können, müssen aber nicht, mit Zeilennummern versehen werden. MS-BASIC benötigt intern diese Zeilennummern nur noch für die Programmzeilen, die Ziel eines Programmsprungs (GOTO) sind. Nur die wenigsten Zeilen eines Programms sind aber normalerweise Ziel eines Programmsprungs. Diese Zeilen können mit einer Nummer versehen werden. Noch besser ist es aber, sie mit einer lesbaren Marke zu versehen (siehe unten).

- Programm-Marken: Wichtige Stellen eines Programms, zum Beispiel die Ziele von Sprunganweisungen, können mit einem lesbaren Namen, einer sogenannten Marke (engl. *Label*), versehen werden. Diese wichtigen Stellen werden so deutlich hervorgehoben und das Programm in überschaubare Abschnitte gegliedert.

- Anweisungs-Blocks: Eine beliebig lange Folge von Zeilen eines Programms kann zu einem Block zusammengefaßt werden. Solche Blöcke finden zum Beispiel bei bedingten Anweisungen (IF-THEN-ELSE-ENDIF), Programmschleifen (FOR-NEXT, WHILE-WEND) oder Unterprogrammen (SUB-END-SUB) Anwendung. Die Verwendung von unübersichtlichen Programm-Sprüngen (GOTO-Anweisungen) für diese Zwecke ist nicht mehr nötig.

- Lange Namen: Alle Bezeichnungen für Variablen, Marken, Unterprogramme und so weiter können lange Namen sein (denen man wirklich etwas entnehmen kann) und so jedes Programm wesentlich lesbarer machen. Alle diese Namen können bis zu 40 Buchstaben lang werden, und alle 40 Buchstaben werden auch zur Unterscheidung der verschiedenen Namen eingesetzt.

- Reiche Auswahl an Datentypen: Für Variablen und die Ergebnisse von Funktionen stehen insgesamt 5 verschiedene Datentypen zur Verfügung. Amiga-BASIC kennt zwei Genauigkeitsstufen für ganze Zahlen (16 Bit und 32 Bit), zwei Genauigkeitsstufen für Fließpunktzahlen (32 Bit und 64 Bit) und Strings (Texte) die bis zu 32 768 Buchstaben enthalten können. Arrays (Felder bzw. Matrizen) können bis zu 255 Dimensionen haben und in jeder Dimension bis zu 32 768 Elemente enthalten.

- Subprogramme: Neben den auch aus anderen BASIC-Dialekten bekannten Subroutinen gibt es auch echte »Subprogramme« oder »Prozeduren«, die die Übersichtlichkeit großer Programme wesentlich verbessern und ihre Fehleranfälligkeit verringern können. Subprogramme haben stets einen (hoffentlich aussagekräftigen) Namen und können Parameter besitzen, die beim Aufruf mit einem Wert belegt werden. Weiterhin können Subprogramme eigene (lokale) Variablen besitzen, die von denen des Hauptprogramms verschieden sind, selbst wenn sie genauso heißen. Subprogramme können sich sogar selbst aufrufen, was gar nicht so selten oder überflüssig ist, wie man vielleicht meint (der Informatiker nennt solche Subprogramme dann »rekursiv«).

- Programm-Chaining: Jedes BASIC-Programm kann andere BASIC-Programme aufrufen und diesen sogar (über die COMMON-Anweisung) Daten übergeben. Dies nennt man Programm-Verkettung oder -Chaining. Auf diese Weise kann man sehr große Programme entwickeln, von denen aber immer nur die gerade benötigten Teile im Speicher sind. So bleibt mehr Platz für Daten frei.

- Komfortable Dateiverwaltung: MS-BASIC besitzt reichhaltige Möglichkeiten für die Verarbeitung von Dateien, die auf der Diskette oder einer Festplatte gespeichert sind. Solche Dateien können in Datensätze gleicher Struktur aufgeteilt werden, bei denen jeder Datensatz aus einer Reihe von Feldern mit aussagekräftigen Namen besteht. Dateien können sequentiell (also Datensatz für Datensatz) oder auch mit wahlfreiem Zugriff (engl. *random access*) auf jeden beliebigen Datensatz bearbeitet werden.

16.3 Wie Amiga-BASIC die Fähigkeiten des Amiga 500 nutzt

Obwohl Amiga-BASIC sehr ähnlich zu den anderen MS-BASIC-Dialekten auf anderen Computern ist, hat Microsoft es aber doch auch sehr sorgfältig an die speziellen Fähigkeiten des Amiga angepaßt. Fast jede dieser Fähigkeiten, die den Amiga 500 ja erst zu einem so außergewöhnlichen Computer machen, kann von BASIC aus genutzt werden. Es ist also problemlos möglich, eigene Programme zu schreiben, die wie »richtige Amiga-Programme» aussehen; mit eigenen Screens, Fenstern, Menüs und Maus-Bedienung. Zugleich werden diese Möglichkeiten aber auch für die Programmentwicklung selbst genutzt und machen dem Programmierer so das Leben etwas leichter.

16.3.1 Die Bedienung von Amiga-BASIC

Amiga-BASIC ist, wie die meisten BASIC-Systeme, ein sogenannter Interpreter. Dies bedeutet, daß für die Ausführung eines Programms immer ein kleines Programm (eben der

Interpreter) benötigt wird, das Zeile für Zeile durch den Programmtext läuft, jeweils eine Anweisung liest und sie dann ausführt. Dies ist ein verhältnismäßig (für einen Computer) langsamer Prozeß. Schnellere Programme erhält man meist durch einen Compiler, der einen Programmtext nimmt und in einen Code übersetzt, der direkt von der CPU des Computers ausgeführt werden kann, den sogenannten Maschinen-Code. Auch für Amiga-BASIC wird sicherlich über kurz oder lang ein solcher Compiler auf den Markt kommen.

Die Arbeit mit einem Interpreter hat aber unschätzbare Vorteile für den Bedienungskomfort. Während die oben beschriebene Übersetzung eines Programms in Maschinen-Code ein langwieriger und vor allem für Anfänger fehleranfälliger Vorgang ist, braucht man ein Amiga-BASIC-Programm einfach nur einzutippen und kann es sofort starten. Auch jede Änderung kann sofort – ohne jede Wartezeit – ausprobiert werden.

Weiterhin arbeitet man mit Amiga-BASIC in zwei verschiedenen Fenstern. In einem, dem sogenannten Output-Fenster, spielen sich die Ein- und Ausgaben von Programmen ab. Sie können hier auch über die Tastatur Befehle an das BASIC-System erteilen. Das zweite Fenster, das sogenannte »Listing-Fenster«, enthält den Text des gerade bearbeiteten Programms. BASIC-Programme können darüber hinaus beliebig viele zusätzliche Fenster öffnen, für deren Inhalt sie dann auch selbst zuständig sind. Aber auch die beiden ersten Fenster müssen nicht immer sichtbar sein. Wenn ein Programm fehlerfrei läuft, wird man üblicherweise das Listing-Fenster schließen und Programme, die selbst eigene Fenster öffnen, werden oft auch das erste Output-Fenster schließen und alle Ein- und Ausgaben in den eigenen Fenstern erledigen.

Auch die Eingabe und das Ändern von Programmen im Listing-Fenster ist sehr komfortabel. Wer BASIC-Interpreter auf anderen Computern kennt, wird staunen. Zur Bearbeitung des Programmtextes im Listing-Fenster stehen Ihnen ähnliche Möglichkeiten wie im *Notepad* zur Verfügung. Mehr dazu aber weiter unten.

Tauchen während des Programmablaufs Fehler auf, so stoppt BASIC, gibt eine entsprechende Fehlermeldung aus und zeigt Ihnen die Fehlerstelle auch durch einen dünnen Rahmen um die entsprechende Stelle im Programmtext. War das Listing-Fenster geschlossen, wird es hierzu zuvor geöffnet. Sie können den Fehler nun beseitigen und das Programm noch einmal laufen lassen.

BASIC selbst findet aber natürlich nur recht grobe Fehler, zum Beispiel Tippfehler. Die meisten logischen Fehler müssen Sie auf andere Weise herausfinden, wofür es aber eine ganze Reihe von Hilfen gibt.

Mit dem Befehl *Trace On* haben Sie zum Beispiel eine genaue Kontrolle über den Ablauf eines Programms. Nachdem *Trace On* ausgewählt wurde, zeigt Amiga-BASIC während der Ausführung eines Programms kontinuierlich an, in welcher Zeile sich das Programm gerade befindet. Hierzu wird im Listing-Fenster die gerade ausgeführte Zeile orange umrahmt. Und mit dem Befehl *Step* schließlich können Sie Schritt für Schritt (Zeile für Zeile) durch ein Programm gehen. Nach der Ausführung jeder Anweisung des Programms hält Amiga-BASIC automatisch an.

16.3.2 Unterstützung der Amiga-Fähigkeiten

Amiga-BASIC sieht eine ganze Reihe von Funktionen vor, die es erlauben, von einem BASIC-Programm aus alle wichtigen Elemente der Amiga-Benutzerschnittstelle Intuition zu verwenden. Dies ist natürlich wichtig für jeden, der Programme schreiben möchte, die wie »echte Amiga-Programme aussehen«, also sich von ihrer Bedienung her nicht von professionellen Programmen unterscheiden.

Im folgenden finden Sie einen kleinen Überblick über diese Möglichkeiten. Er ist wieder recht stichwortartig gehalten und soll Ihnen nur einen Eindruck von den Möglichkeiten verschaffen, die Amiga-BASIC bietet. Detaillierte Erläuterungen der einzelnen Befehle finden sich im Kapitel 8 des Amiga-BASIC-Handbuchs.

Grafik

Der einfachste Grafik-Befehl von Amiga-BASIC ist der LINE-Befehl. Mit ihm können Linien zwischen zwei Punkten in einer wählbaren Farbe gezogen werden. Alternativ kann er auch dazu verwendet werden, Rechtecke zu zeichnen, die auf Wunsch gleich mit einer bestimmten Farbe gefüllt werden. Der Befehl CIRCLE kann Kreise, Ellipsen sowie Kreisbögen mit einem bestimmten Radius um einen festgelegten Mittelpunkt herum zeichnen. Auch diese können auf Wunsch gleich gefüllt werden. Komplexere Formen, Polygone (Vielecke), können mit dem AREA-Befehl Punkt für Punkt definiert und mit AREAFILL dann gezeichnet werden.

Noch komplexere Formen als die oben aufgeführten muß man aus diesen »Elementen« zusammensetzen. Sobald eine Fläche der gewünschten Form aber erst einmal einen Rahmen hat, kann sie mit Befehl PAINT jederzeit mit einer beliebigen Farbe gefüllt werden. Dies wirkt genauso wie die Anwendung des Farbeimers in Graphicraft oder Deluxe Paint.

Bei den Farben, die für eine BASIC-Grafik verwendet werden können, ist man natürlich nicht auf die vier »Workbench-Farben» beschränkt. Mit dem Befehl PALETTE kann man diese jederzeit ändern. Über PALETTE kann man jedes der Farbregister im Video-Chip Denise mit neuen Rot-, Grün- und Blau-Werten füllen. Wem vier Farben nicht genügen, der kann sich mit SCREEN-Anweisung (siehe unten) einen eigenen Screen erzeugen, der bis zu 32 Farben enthalten kann.

Fenster und Screens

»Normalerweise« gehen alle Ausgaben eines Programms in das eine Output-Fenster, das gleich am Anfang, wenn man BASIC startet, erscheint. Auch die eben beschriebenen Grafikbefehle arbeiten zunächst in diesem Fenster. Wem ein Fenster aber nicht genügt oder das Standard-Output-Fenster nicht gefällt, der kann sich mit dem Befehl WINDOW ein neues erzeugen, dessen Titel, anfängliche Größe und sonstige Eigenschaften völlig frei gewählt werden können. Auch Fenster, die von Programmen erzeugt wurden, können als Output-Fenster verwendet werden, indem man sie mit der WINDOW OUTPUT-Anweisung zum aktuellen Output-Fenster macht. Man kann zum Beispiel PRINT-Befehle in diese Fenster schicken und in ihnen Grafiken zeichnen. Von einem Programm erzeugte Fenster, aber auch die beiden Standard-Fenster, können jederzeit auch von einem Programm aus mit WINDOW CLOSE wieder geschlossen werden.

Für anspruchsvolle Grafiken genügt es aber kaum, ein spezielles Fenster zu öffnen. In BASIC steht einem ja »normalerweise» nur ein Screen mit vier Farben zur Verfügung. Braucht man mehr Farben, eine andere Bildschirmauflösung oder möchte man gar den HAM-Modus des Denise-Chips einmal ausprobieren, kann man mit SCREEN einen neuen virtuellen Bildschirm erzeugen und ihn mit SCREEN CLOSE wieder schließen, sobald er nicht mehr gebraucht wird.

Menüs

Die Pull-down-Menüs sind eine der wichtigsten Komponenten der Amiga-Benutzerschnittstelle Intuition. Auch von Amiga-BASIC aus können deshalb neue Menüs und Menü-Punkte erzeugt werden. Die BASIC-Anweisung MENU tut genau dies. Amiga-BASIC kann allerdings nur reine Text-Menüs erzeugen. Grafische Menüs, wie die Palette oder die Hilfsmittelauswahl in Graphicraft, sind nicht möglich.

Mit einer zweiten Form des MENU-Befehls kann ein Programm jederzeit feststellen, aus welchem Menü der Benutzer zuletzt welchen Befehl ausgewählt hat. Sie werden in den folgenden Abschnitten noch anhand einiger Beispiele sehen, wie man das macht. Zum Feststellen der Tatsache, ob der Benutzer überhaupt eine Menü-Auswahl getätigt hat, wird ein anderer Mechanismus verwendet, der weiter unten noch erläutert wird.

Die Maus

Einen Teil der Mausbewegungen und Mausklicks überwacht und interpretiert ja bereits die Amiga-Benutzerschnittstelle Intuition (zum Beispiel das Bewegen von Fenstern und Herunterholen von Pull-down-Menüs mit der Maus). Mausklicks innerhalb eines Fensters müssen aber normalerweise von dem für dieses Fenster zuständigen Programm behandelt werden. Ein BASIC-Programm verwendet dafür üblicherweise die Funktion MOUSE. Mit dieser kann man feststellen, ob die linke Maustaste im Moment gedrückt ist oder nicht, über welchem Punkt sie gedrückt und wieder losgelassen wurde und ob es ein Doppelklick war oder nicht.

Die rechte Menütaste der Maus kann von BASIC aus nicht abgeprüft werden. Sie kann nur indirekt über die MENU-Funktion von einem BASIC-Programm benutzt werden.

BOBs und Sprites

Eine der faszinierendsten Möglichkeiten, die der Amiga grafisch bietet, sind ja bewegte Objekte. Diese gibt es dabei in zwei »Geschmacksrichtungen»: die in Hardware erzeugten Sprites, die schnell sind, aber diversen Einschränkungen unterliegen, oder die etwas langsameren, aber wesentlich vielseitigeren BOBs (Blitter Objects), die von der Systemsoftware des Amiga und dem Blitter ermöglicht werden. In Amiga-BASIC werden beide Typen einfach Objekte genannt und immer mit demselben Satz von Befehlen behandelt.

Für die Definition des Aussehens von Objekten befindet sich ein separates Programm namens Objedit auf der Extras-Diskette. Es ist übrigens in BASIC geschrieben und demonstriert so recht eindrucksvoll die Möglichkeit, sinnvolle Amiga-Programme in BASIC zu entwickeln.

Die mit diesem Programm entwickelte Form eines Objektes kann dann mit OBJECT.SHAPE für die Erzeugung eines neuen Sprites oder BOBs verwendet werden. Ein solches Objekt kann

man dann mit OBJECT.X und OBJECT.Y an einem bestimmten Punkt am Bildschirm zeigen und ihm mit OBJECT.VX und OBJECT.VY eine Geschwindigkeit geben, mit der es sich (ohne weiteres Zutun des Programms!) über den Bildschirm bewegt. Mit OBJECT.AX und OBJECT.AY bekommt es dann sogar noch eine Beschleunigung, wodurch die Geschwindigkeit des Objekts sich mit der Zeit automatisch ändert. Treffen zwei Objekte aufeinander oder versuchen sie einen »zulässigen» Bereich zu verlassen, kann sich ein Programm davon informieren lassen (siehe auch den nächsten Abschnitt) und mit dem Befehl COLLISION feststellen, welches Objekt auf welches Hindernis gestoßen ist.

Mit OBJECT.PRIORITY kann schließlich auch noch festgelegt werden, welches Objekt »vor» welchem anderen erscheint, wenn zwei oder mehr Objekte an ein und derselben Stelle am Bildschirm stehen. 3-D-Effekte sind so auch für ein BASIC-Programm kein Problem.

In einem BASIC-Programm können beliebig viele bewegte Objekte verwendet werden, solange der freie Speicherplatz ausreicht (BOBs benötigen teilweise recht viel Speicherplatz).

Abfangen von Ereignissen

Auf viele Ereignisse muß ein Programm sehr schnell reagieren können. Wenn zwei Objekte zusammenstoßen oder die linke Maus-Taste niedergedrückt wird, muß am besten sofort eine Reaktion erfolgen. Dies ist aber, wenn es sehr viele solche Ereignisse gibt, die ja alle regelmäßig und möglichst schnell abgeprüft werden müssen, schwer zu programmieren.

Amiga-BASIC bietet deshalb mit dem ON...GOSUB-Befehl (die drei Pünktchen deuten ein anderes Wort an, das hier eingesetzt werden muß) eine Möglichkeit, ein bestimmtes Ereignis mit einer bestimmten Subroutine fest zu verknüpfen. Dies nennt man dann »Abfangen des Ereignisses« (engl. *Event Trapping*). Immer wenn dann ein solches Ereignis, zum Beispiel ein Mausklick, eintritt, ruft Amiga-BASIC sofort diese Subroutine auf, die in der entsprechenden ON...GOSUB-Anweisung genannt wurde – egal welcher andere Programmteil gerade aktiv ist. Wenn die Subroutine mit RETURN beendet wird, springt das Programm automatisch wieder an die Stelle zurück, an der es unterbrochen wurde. Eine sehr schnelle Reaktion auf bestimmte Ereignisse kann so stets gewährleistet werden.

Die Ereignisse, die auf diese Weise abgefangen werden können, sind:

- Die Auswahl eines Menü-Punktes (ON MENU)
- Ein Mausklick (ON MOUSE)
- Das Ablaufen einer bestimmten Zeitspanne (ON TIMER)
- Die Kollision zweier Objekte (ON COLLISION)
- Der Versuch des Benutzers, ein Programm zu stoppen (ON BREAK)
- Ein Programmfehler (ON ERROR)

Die Subroutine, die auf ein bestimmtes Ereignis hin von Amiga-BASIC aufgerufen wird, kann jederzeit mit einem erneuten Aufruf von ON...GOSUB wieder geändert werden. Je nach Programmsituation können also andere Programmteile ein bestimmtes Ereignis bearbeiten.

Musik- und Sprachausgabe

Und *last, but not least* können auch die musikalischen und sprachlichen Fähigkeiten des Amiga von BASIC aus genutzt werden. Der Befehl SOUND gibt eine Note einer bestimmten Frequenz mit einer bestimmten Dauer und Lautstärke auf einem der vier Klangkanäle aus. Für jeden Klangkanal kann zuvor eine Klangform mit dem Befehl WAVE festgelegt werden.

Die Befehle TRANSLATE und SAY schließlich dienen der Sprach-Synthese. TRANSLATE wandelt einen (englischen) Text in die Phonemschreibweise um und SAY gibt einen Text in Phonemschreibweise aus. Um wirklich gute Ergebnisse (oder deutsche Sprache) zu erzielen, wird man die Phonemschreibweise eines Wortes im allgemeinen nicht durch TRANSLATE ermitteln lassen, sondern mit einem guten Wörterbuch feststellen.

16.4 Umgang mit Amiga-BASIC

In diesem und den folgenden Abschnitten werden Sie Amiga-BASIC etwas näher kennenlernen – sofern Sie Lust dazu verspüren. Dazu werde ich zunächst den Umgang mit dem Amiga-BASIC-Interpreter kurz vorstellen und dann einige kleine Beispielprogramme vorstellen.

16.4.1 Wie Sie Amiga-BASIC starten

Amiga-BASIC ist ein Werkzeug wie alle anderen Workbench-Werkzeuge, die Sie schon kennengelernt haben. Das entsprechende Piktogramm befindet sich auf der Extras-Diskette. Sie können Amiga-BASIC starten, indem Sie einfach einen Doppelklick auf dieses Piktogramm machen.

Danach erscheinen zwei Fenster am Bildschirm. Sie heißen *BASIC* und *LIST*. BASIC ist das Output-Fenster und LIST das Listing-Fenster. Wenn Sie nun die Menütaste drücken und in der Titelleiste des Bildschirms hin- und herfahren, können Sie auch die Menüs von Amiga-BASIC sehen. Verlassen können Sie Amiga-BASIC übrigens mit dem Befehl *Quit* im Menü *Project* – aber darauf wären Sie als erfahrener Amiga-Besitzer sicherlich auch selbst gekommen.

16.4.2 Wie Sie Programme eingeben und editieren

Das Listing-Fenster dient zur Eingabe und zur Änderung von BASIC-Programmen. (BASIC-Anweisungen, die Sie im Output-Fenster eingeben, werden hingegen sofort ausgeführt, wenn Sie [Return] drücken und können nicht wieder geändert werden.) Das Listing-Fenster verhält sich fast genauso wie das Fenster des Programms *Notepad*, in dem Sie ja auch Texte editieren können. Ein blinkender Strich, die Schreibmarke, markiert immer die Stelle, an der neu eingegebene Buchstaben erscheinen. Tippen Sie einfach einmal ein paar Zeilen ein. Diese müßten daraufhin im Listing-Fenster erscheinen. Mit der [Backspace]-Taste können Sie dann wie üblich den Buchstaben links von der Schreibmarke und mit [Del] den rechts von der Schreibmarke löschen.

Die Schreibmarke (der blinkende Strich) kann mit der Maus oder mit den vier Cursor-Tasten beliebig im ganzen Text eines Programms verschoben werden. Ein Verschieben des Fensters über ein großes Programm geschieht jedoch nicht mit Rollbalken, wie es in anderen Programmen üblich ist, sondern ebenfalls mit den Cursor-Tasten (indem man die [Shift]-Taste festhält und dann auf die Cursor-Taste drückt, deren Aufdruck in die Richtung weist, in die man das Fenster verschieben möchte).

Wenn Sie eine beliebige Stelle im Text mit der linken Maustaste anklicken und bei gedrückter Taste den Mauszeiger zu einer anderen Stelle ziehen, werden alle Buchstaben zwischen diesen beiden Punkten hervorgehoben. (Einzelne Wörter können auch durch einen Doppelklick über einer beliebigen Stelle des Wortes hervorgehoben werden.) Ein so ausgewählter Bereich kann mit den Menü-Befehlen aus dem Menü *Edit* dann bearbeitet werden. Der Befehl *Cut* entfernt ihn aus dem Text, legt ihn aber zugleich in eine »Zwischenablage«. Der Befehl *Copy* kopiert den hervorgehobenen Text nur in die Zwischenablage. Und der Befehl *Paste* fügt schließlich den Text aus der Zwischenablage vor der Schreibmarke in den Programmtext ein. Diese drei Befehle verhalten sich sehr ähnlich wie die gleichnamigen Befehle im *Notepad*. Mit Hilfe des *Edit*-Menüs kann man sehr leicht auch größere Umstellungen im Programm vornehmen.

Ein großer Vorteil von Amiga-BASIC gegenüber primitiveren Versionen dieser Sprache sind die Möglichkeiten zur Formatierung eingegebener Programme. So kann das Innere von Schleifen oder IF-THEN-ELSE-Anweisungen nach rechts eingerückt werden, wodurch solche Blöcke natürlich gleich viel mehr ins Auge stechen. Die [Tab]-Taste ist ideal für solche Einrückungen. Sie fügt jeweils 3 Leerzeichen bei der Schreibmarke ein.

Neben diesen Formatierungen, für die man selbst sorgen muß, gibt es aber auch automatische Formatierungen, die Amiga-BASIC selbst durchführt. Wenn Sie ein BASIC-Schlüsselwort, wie zum Beispiel IF, FOR oder GOTO eingegeben haben und in eine neue Zeile gehen, schreibt BASIC diese Schlüsselwörter automatisch groß, egal, wie Sie sie eingegeben haben. Dies erhöht natürlich ebenfalls die Lesbarkeit des Programms und hilft Ihnen zugleich, Tippfehler zu entdecken. Wollten Sie nämlich ein Schlüsselwort eingeben und es erscheint nicht in Großbuchstaben, muß sich darin ein Tippfehler befinden. Verwenden Sie aber einen Variablen- oder Subprogramm-Namen und dieser erscheint plötzlich in Großbuchstaben, obwohl Sie ihn klein eingegeben haben, so gibt es ein Schlüsselwort gleichen Namens, woraufhin Sie sich einen anderen Namen ausdenken sollten. Sie werden diese Effekte gleich beobachten können. Geben Sie dazu einfach das folgende kurze Programm im Listing-Fenster ein (löschen Sie dazu eventuell die Spuren Ihrer ersten Tipp-Versuche):

```
Start:
print "Hallo Ihr da!"
goto start
```

Wenn Sie dieses Programm eingegeben haben ([Return] am Zeilenende nicht vergessen!), sollte es so aussehen wie das folgende Programm:

```
start:
PRINT "Hallo Ihr da!"
GOTO start
```

Amiga-BASIC hat die beiden Schlüsselwörter *print* und *goto* automatisch groß geschrieben und auch dafür gesorgt, daß die beiden Vorkommnisse des Wortes *start* identisch geschrieben werden.

16.4.3 Wie Sie Programme starten und wieder anhalten

Sie können dieses winzige Programm nun starten, indem Sie den Befehl *Start* aus dem Menü *Run* wählen. Das Output-Fenster kommt daraufhin nach vorn und eine endlose Kette von »Hallo Ihr da!« wird darin ausgegeben. Das Programm hat kein »natürliches« Ende, da der Sprung zurück nach *start* in der letzten Zeile eine Endlosschleife schließt. Sie müssen es deshalb »von außen« anhalten. Wählen Sie hierzu den Befehl *Stop* aus dem Menü *Run*. Das Programm hält dann an und das Listing-Fenster kommt automatisch wieder nach vorn.

16.4.4 Wie Sie Fehler in Programmen suchen

Bauen Sie nun einmal einen kleinen Fehler in das Programm ein, um zu sehen, wie Amiga-BASIC darauf reagiert. Fügen Sie dazu vor GOTO start in eine neue Zeile den Befehl GOTO Hallo ein. Das Programm müßte dann aussehen wie folgt:

```
start:
PRINT "Hallo Ihr da!"
GOTO Hallo
GOTO start
```

Wenn Sie nun dieses Programm starten, ertönt sofort ein kleiner Piepston und ein schmales Fenster erscheint am oberen Bildschirmrand, das Ihnen die Fehlermeldung *Undefined label* (nicht definierte Sprungmarke) zeigt. Sie können dieses Fenster mit einem Klick auf den OK-Knopf darin schließen. Unterdessen hat Amiga-BASIC aber auch schon das Listing-Fenster nach vorne gebracht und die Zeile mit dem Fehler darin markiert. Der Fehler selbst ist Ihnen ja wahrscheinlich klar, oder? Sie können mit der GOTO-Anweisung nicht zu einer Sprungmarke springen, wenn diese Sprungmarke nirgends definiert wurde. Die Sprungmarke *start* wurde ja definiert, indem Sie den Namen *start* mit einem Doppelpunkt dahinter an den Anfang einer Zeile geschrieben haben. Löschen Sie deshalb nun den fehlerhaften Befehl *GOTO Hallo* wieder und starten Sie das Programm erneut. Nun müßte wieder die endlose Folge von »Hallo Ihr da!«-Zeilen erscheinen.

Rufen Sie nun einmal den Befehl *Trace On* aus dem Menü *Run* auf. Das Programm wird dann merklich langsamer. Ansonsten passiert aber scheinbar nichts. Daß nichts passiert liegt daran, daß das Listing-Fenster vom Output-Fenster verdeckt wird. Holen Sie es deshalb nun – bei laufendem Programm – mit dem Befehl *Show List* im Menü Windows nach vorn. Nun können Sie verfolgen, was geschieht, wenn Sie *Trace On* aufrufen. Während des Ablaufs des Programms wird der gerade ausgeführte Befehl immer mit einem Rahmen versehen. Sie können damit sehr gut verfolgen, ob das Programm wirklich die »Wege geht«, die Sie geplant haben. In diesem simplen Programm ist das natürlich keine große Hilfe. In einem großen komplexen Programm kann es entscheidend dazu beitragen, logische Fehler zu finden.

Sie können die Markierung der aktiven Zeile jederzeit wieder abschalten, indem Sie *Trace Off* aus dem Menü *Run* wählen. Das Programm arbeitet dann wieder mit gewohnter Geschwindigkeit. Wenn Ihnen aber selbst die Geschwindigkeit bei *Trace On* noch zu groß ist, haben Sie noch eine andere Möglichkeit, den Programmfluß zu kontrollieren. Halten Sie dazu das Programm zunächst einmal mit dem Befehl *Stop* aus dem Menü *Run* an und wählen Sie dann den Befehl *Step* aus dem Menü *Run*. Nun wird auch wieder der aktuelle Befehl dünn umrahmt, das Programm läuft aber nicht weiter. Erst, wenn Sie wieder *Step* aufrufen, macht es den nächsten Arbeitsschritt. Dazu einen Menübefehl aufrufen zu müssen, ist natürlich umständlich. Die Tastenkombinationen [*A*]+[T], das Tastaturäquivalent von *Step*, ist da schon handlich. Halten Sie dazu die Amiga-Taste fest und drücken Sie immer dann auf [T], wenn Sie wollen, daß das Programm einen Schritt weitergeht. Das ist praktisch, nicht wahr?

Wenn es Ihnen aber auf den Geist geht, für jeden Programmschritt auf eine Taste drücken zu müssen, können Sie den Befehl *Continue* aus dem Menü *Run* wählen. Das Programm läuft dann normal weiter. Der letzte Befehl im *Run*-Menü, *Suspend,* wirkt ähnlich wie *Stop*. Er hält also das laufende Programm an. Nur wird hierbei das Programm nur so lange unterbrochen, bis auf eine beliebige Taste gedrückt wird.

16.4.5 Wie Sie Programme abspeichern

Für Amiga-BASIC gilt dasselbe wie für die meisten anderen Programme: Neueingaben oder Änderungen (an einem Programm) werden zunächst einmal nur in einem Puffer im RAM-Speicher ausgeführt. Wenn Sie die Änderungen auch nach dem Verlassen des Programms erhalten wollen, müssen Sie sie auf Diskette oder Festplatte sichern. Zum Sichern eines BASIC-Programms gibt es die zwei Befehle *Save* und *Save As* im Run-Menü.

Wenn Sie ein neues Programm eingeben und dann *Save* aufrufen, erscheint ein Requester, der Sie nach dem Namen des Programms fragt. Klicken Sie dann einmal in das Textgadget unter der Zeile *Save program as:* und geben Sie den Namen ein, unter dem das Programm gesichert werden soll. Wenn es in einer bestimmten Schublade abgelegt werden soll, müssen Sie hier den vollen Pfadnamen eintragen (also zum Beispiel *ExtrasD:BasicDemos/test*, um das Programm auf der Diskette *ExtrasD* in der Schublade *BasicDemos* unter dem Namen *test* abzuspeichern). Sobald Sie auf [Return] drücken oder den OK-Knopf anklicken, wird das Programm dann unter dem angegebenen Namen gesichert. Wenn Sie später – nach Änderungen – das Programm erneut sichern wollen, genügt es, den Befehl *Save* auszuwählen. Ein Name wird dann nicht noch einmal abgefragt.

Wollen Sie das Programm jedoch unter einem anderen Namen abspeichern, so können Sie den Befehl *Save As* verwenden. Er verhält sich genauso wie der erste *Save*-Befehl, erlaubt es Ihnen aber, auch ein Programm, das schon einen Namen hat, noch einmal unter einem anderen zu sichern.

16.4.6 Wie Sie Programme einlesen

Das Einlesen eines BASIC-Programms geht ähnlich vonstatten wie das Abspeichern. Wählen Sie dazu den Befehl *Open* aus dem Menü *Run*. Wenn Sie das aktuelle Programm zuvor noch

nicht gesichert hatten, erscheint nun eine Warnung (*Current program is not saved, Do you want to save it, before proceeding?*) in Form eines Requesters mit drei Knöpfen. Klicken Sie in diesem Requester auf den Knopf *Yes*, wird das aktuelle Programm gesichert, bevor das neue geladen wird. Klicken Sie auf *No*, werden die Änderungen am aktuellen Programm verworfen. Und wenn Sie auf *Cancel* klicken, wird der Befehl abgebrochen.

Wenn Sie *Yes* oder *No* angeklickt haben oder kein Programm zu sichern war, erscheint dann ein Requester, in dem Sie den Namen des zu ladenden Programms eingeben müssen. Klicken Sie dazu das Textgadget unter der Zeile *Name of program to load:* an und tippen Sie den Pfadnamen des gewünschten Programms ein. Sobald Sie auf [Returnl drücken oder den OK-Knopf anklicken, wird dieses Programm geladen. Falls es kein Programm dieses Namens gibt, erscheint eine Fehlermeldung (*File not found*).

Sowohl bei *Save* wie auch bei *Open* können Sie sich die Angabe des vollen Pfadnamens sparen, wenn das Programm, das abgespeichert oder geladen werden soll, im aktuellen Directory liegt. Beim Start entspricht das aktuelle Directory der Schublade, in der Sie den Doppelklick gemacht haben, um Amiga-BASIC zu starten. Das ist die Schublade mit dem Piktogramm *AmigaBASIC*, oder die mit dem Piktogramm eines Programms, wenn Sie Amiga-BASIC durch einen Doppelklick auf ein BASIC-Programm gestartet haben. Sie können aber auch innerhalb von Amiga-BASIC das aktuelle Directory ändern. Klicken Sie dazu einmal (während kein Programm läuft) in das Output-Fenster und geben Sie den Befehl *chdir* und dahinter den Namen des gewünschten Directory in Anführungsstrichen ein. Der folgende Befehl macht zum Beispiel das Directory *BasicDemos* auf der Diskette *ExtrasD* zum aktuellen Directory:

```
chdir "extrasd:basicdemos"
```

Programme aus diesem Directory (dieser Schublade) können Sie nun laden, indem Sie einfach nur ihren Namen (und nicht den vollständigen Pfadnamen) in den entsprechenden Requester eintragen und [Return] drücken. Welche Programme in diesem Directory zur Auswahl stehen, können Sie feststellen, indem Sie wieder im Output-Fenster den folgenden Befehl eingeben:

```
files
```

Daraufhin erscheint eine Liste der Dateien in diesem Directory. Wenn Sie Lust darauf verspüren, ein richtig großes Programm kennenzulernen, können Sie ja nun einmal (mit *Open*) ein paar dieser Programme aus der Schublade *BasicDemos* der Extras-Diskette laden, laufen lassen und sich im Listing-Fenster anschauen. Sehr interessant sind zum Beispiel die Programme *Dreier*, *Music* und *Screen*.

16.5 Grafik mit Amiga-BASIC

Nach diesem kleinen Schnellkurs über den Umgang mit Amiga-BASIC sollten Sie nun – sofern Sie Lust dazu verspüren – aber auch anhand einiger kleiner Beispielprogramme ein paar BASIC-Befehle ausprobieren. Den Anfang machen in diesem Kapitel die Grafik-Befehle.

16.5.1 Pixel

Der einfachste Grafik-Befehl ist PSET (***Pixel Set***). PSET färbt genau einen Bildschirmpunkt (ein Pixel) auf dem Bildschirm ein. Er benötigt zwei Parameter, eine Koordinatenangabe und eine Zahl, die besagt, welche Farbe dieser Punkt bekommen soll. Koordinatenangaben sind dabei immer zwei Zahlen (eine X- und eine Y-Koordinate), die von einem Komma getrennt in runden Klammern stehen. *PSET (100,100),3* färbt im Output-Fenster zum Beispiel den Punkt an der Position (100,100) mit der Farbe 3 (das ist Orange, falls Sie die Workbench-Farben nicht verändert haben). Das folgende kleine Programm füllt das ganze Output-Fenster mit einer zufällig verteilten Wolke solcher farbiger Pixel an.

```
start:
x = RND * 630
y = RND * 200
c = RND * 3 + 0.4
PSET (x,y),c
GOTO start
```

Der Aufbau des Programms ist denkbar simpel. Es verwendet vor allem die Funktion RND, die eine Zufallszahl zwischen 0 und 1 ergibt. Durch Multiplikation wird dieser Bereich zwischen 0 und 1 dann auf die Bereiche von 0 bis 630, von 0 bis 200 und von 0,4 bis 3,4 »gestreckt«. Diese gestreckten Zufallszahlen werden dann Variablen zugewiesen und schließlich in der PSET-Anweisung dazu verwendet, um einen Punkt im Output-Fenster an einer zufälligen Position mit einer zufälligen Farbe zu versehen. Die Farbe *c* liegt dazu in einem Bereich zwischen 0,4 und 3,4. Diese krummen Zahlen rundet der PSET-Befehl selbst auf die vier Zahlen 0, 1, 2 und 3, die die vier Farben auswählen, die auf dem Workbench-Screen zur Verfügung stehen. Das Ganze ist dann in eine Endlos-Schleife gepackt, die Sie mit dem Befehl *Stop* abbrechen müssen, wenn Sie genug davon haben. Das folgende Programm leistet übrigens genau dasselbe, benutzt dazu aber statt des altmodischen GOTOs die elegante WHILE-Schleife.

```
WHILE 1=1
  x = RND * 630
  y = RND * 200
  c = RND * 3 + 0.4
  PSET (x,y),c
WEND
```

Diese WHILE-Schleife zwischen WHILE und WEND läuft so lange, wie die Bedingung hinter WHILE zutrifft (genauer gesagt läuft sie so lange, wie dieser Ausdruck ein Ergebnis ungleich Null ergibt). Und die Bedingung 1=1 trifft immer zu. Dies ist noch ein recht triviales Beispiel. Aber Sie sehen vielleicht schon, daß WHILE-Schleifen zusammen mit der Einrückung ein Programm übersichtlicher machen können.

16.5.2 Linien

Eine ganz einfache und sehr interessante Abwandlung des Programms aus dem letzten Abschnitt macht die LINE-Anweisung möglich. Sie arbeitet sehr ähnlich wie die PSET-Anweisung, benötigt vor der Angabe einer Farbe allerdings zwei Koordinaten, die einen Anfangs- und einen Endpunkt bestimmen, zwischen denen dann eine dünne Linie in der angegebenen Farbe gezogen wird. Das folgende Programm enthält die nötigen Änderungen, um statt Punkte nun farbige Linien auf dem Bildschirm erscheinen zu lassen.

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c
WEND
```

(Beachten Sie bitte, daß die beiden Koordinatenangaben in der LINE-Anweisung nicht durch ein Komma, sondern durch einen Bindestrich »–« voneinander getrennt sind!) Diese Programme brauchen Sie nicht mit *Stop* abzubrechen. Es genügt, einmal in das Output-Fenster zu klicken und dann auf eine beliebige Taste zu drücken. Diesen Buchstaben liefert die INKEY$-Funktion dann als Ergebnis zurück. INKEY$ ist dann nicht mehr gleich "" (dem leeren String) und die WHILE-Schleife bricht ab. Ansonsten gibt es an diesem Programm nicht viel Neues. Statt einer Koordinatenangabe erzeugen die Zufallsanweisungen hier nun zwei, da die LINE-Anweisung zwei davon braucht.

16.5.3 Mondrian

Hinter der LINE-Anweisung verbirgt sich jedoch mehr, als auf den ersten Blick ersichtlich ist. Sie haben bislang nur die einfachste Form mit zwei Koordinaten und einer Farbangabe kennengelernt. Dahinter kann jedoch noch ein Parameter kommen. Dieser kann »b« oder »bf« lauten und ändert die Wirkung der LINE-Anweisung total. Statt eine Linie zu ziehen, zeichnet diese Anweisung dann ein Rechteck, dessen obere linke und untere rechte Ecke die beiden Koordinaten angeben. Verwenden Sie den Parameter »b« hinter der Farbangabe, wird dieses Rechteck nur mit einer dünnen Linie gezogen. Verwenden Sie jedoch »bf«, wird das Rechteck mit der angegebenen Farbe gefüllt. Die folgenden beiden Programme benutzen diese Optionen der LINE-Anweisung und erzeugen damit Bilder, die etwas an die Werke des holländischen Künstlers Mondrian erinnern:

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c, b
WEND
```

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c, bf
WEND
```

Falls Ihnen die dabei auftauchenden Rechtecke zu groß sind, können Sie eine weitere Möglichkeit der LINE-Anweisung verwenden, um deren Abmessungen zu begrenzen. Wenn Sie nämlich vor das zweite Koordinatenpaar (x2,y2) das Schlüsselwort STEP schreiben, so geben x2 und y2 nicht mehr die Position des zweiten Eckpunkts, sondern die Abstände des zweiten Eckpunkts vom ersten an. Wenn Sie dann noch dafür sorgen, daß x2 und y2 klein bleiben, bleiben auch die dabei entstehenden Rechtecke klein, wie das folgenden Programm demonstriert:

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 60
  y1 = RND * 200
  y2 = RND * 30
  c = RND * 3 + 0.4
  LINE (x1,y1)-STEP (x2,y2),c, bf
WEND
```

16.5.4 Eigene Fenster und Screens

Obwohl die bisherigen Beispielprogramme schon recht nette Bildchen zustande brachten, fehlt ihnen noch der besondere Pfiff. Das liegt einfach daran, daß Sie zu wenig Farben verwenden. Der Workbench-Screen bietet aber nur vier Farben. Deshalb bleibt uns nichts anderes übrig, als einen neuen Screen zu erzeugen und darin ein neues Output-Fenster zu öffnen. Dazu werden die Anweisungen SCREEN und WINDOW benötigt, die beide recht kompliziert sind und viele Parameter benötigen. Zunächst aber das Programm selbst (und danach die Erläuterungen):

```
SCREEN 1, 320, 256, 5, 1
WINDOW 3, "Mondrian", (0,0)-(310,240),2, 1
WINDOW OUTPUT 3
WHILE INKEY$=""
  x1 = RND * 310
  x2 = RND * 310
  y1 = RND * 230
  y2 = RND * 230
  c = RND * 31 + 0.4
  LINE (x1,y1)-(x2,y2), c, bf
WEND
WINDOW CLOSE  3
WINDOW OUTPUT 1
SCREEN CLOSE  1
```

Am Anfang des Programms wird ein neuer Screen erzeugt. Die Parameter der SCREEN-Anweisung (in dieser Reihenfolge) bedeuten: Nummer des Screens, Breite, Höhe, Tiefe und Modus. Die Nummer wird nur dazu benötigt, wenn man diesen Screen später ansprechen will. Die Bedeutung von Breite und Höhe dürfte klar sein. Die Tiefe bestimmt die Anzahl der möglichen Farben, wie es in Kapitel 12 beschrieben wurde. In diesem Fall ist die Tiefe gleich 5 und es stehen damit 32 Farben zur Verfügung. Der Modus-Parameter bestimmt den Grafik-Modus des Screens. Ein Modus von 1 bedeutet einen Screen niedriger horizontaler Auflösung (320 Punkte) ohne Interlace.

Die zweite Anweisung erzeugt ein neues Fenster. Die Parameter der WINDOW-Anweisung (in dieser Reihenfolge) bedeuten: Fenster-Nummer, Titel, obere linke Ecke, untere rechte Ecke, Fenstertyp und Screen. Die Fenster-Nummer dient wie beim Screen zur Identifikation des Fensters und der Titel ist ein Text, der in der Titelleiste erscheint. Der Fenstertyp-Parameter bestimmt, über welche Eigenschaften das Fenster verfügt. Ein Wert von 2 besagt, daß man dieses Fenster verschieben kann. (Mehr zu den anderen möglichen Werten von Modus finden sie im Amiga-BASIC-Handbuch.) Der letzte Parameter, Screen, bestimmt, in welchem Screen das Fenster auftaucht. In diesem Fall ist es der eben neu erzeugte Screen 1.

Mit WINDOW OUTPUT 3 wird dann das eben erzeugte Fenster Nummer 3 zum aktuellen Output-Fenster, in das Print-Befehle und Grafik-Anweisungen gehen. Hierdurch kommt auch der neue Screen, in dem das Fenster liegt, nach oben. Die darauffolgende Schleife kennen Sie ja bereits aus dem Programm Mondrian im vorigen Abschnitt. Sie können sie wie üblich durch Betätigung einer beliebigen Taste abbrechen.

Ist die WHILE-Schleife beendet, beginnt das Programm, ordnungsgemäß »aufzuräumen«. Hierzu wird das neue Fenster geschlossen (WINDOW CLOSE 3), dann das normale Fenster auf dem Workbench-Screen wieder zum Output-Fenster gemacht und schließlich auch der Screen mit den 32 Farben wieder gelöscht. Das ist alles nicht unbedingt nötig – aber guter Programmierstil.

16.6 Ereignisreiche Programme

Die bislang vorgestellten Beispielprogramme haben – bei aller Farbenpracht – aber nur etwas auf dem Bildschirm gemalt und nicht auf Ihre Eingaben reagiert. Das soll nun anders werden. Und wie sich das für den Amiga gehört, werden wir uns im folgenden nicht mit trivialen Anweisungen wie READ beschäftigen, die es in anderen BASICs für andere Computer auch gibt, sondern um die Eingabe mit der Maus und über Menüs.

16.6.1 Das einfachste Malprogramm der Welt

Damit Sie ein Gefühl für die Benutzung der Maus von einem BASIC-Programm aus bekommen, möchte ich Ihnen nun das einfachste Malprogramm der Welt vorstellen. Sie können damit – zusammen mit etwas Geschick – ohne weiteres kleine Bilder im Outputfenster malen. Trotzdem umfaßt es nur drei Zeilen:

```
WHILE INKEY$=""
  IF MOUSE(0) <> 0 THEN PSET(MOUSE(1),MOUSE(2)),1
WEND
```

Das Programm ist sehr einfach aufgebaut und verwendet fast nur die Funktion MOUSE, die Informationen über den Zustand der Maus liefert. Zunächst wird MOUSE(0) aufgerufen. MOUSE(0) liefert als Ergebnis 0, wenn die linke Maustaste nicht gedrückt ist. Ist die Maustaste gedrückt, oder war sie seit dem letzten Aufruf von MOUSE(0) einmal gedrückt, ergibt MOUSE(0) verschiedene andere Ergebnisse (über die Sie im Amiga-BASIC-Handbuch mehr erfahren können). In diesem Fall wird die Position der Maus erfragt und ein weißer Punkt (Farbe 1) dorthin gesetzt. MOUSE(1) ergibt dazu die horizontale (X-) Position der Maus und MOUSE(2) die vertikale (Y-) Position.

Solange die Maustaste nicht gedrückt ist, tut das Programm nichts – es wartet. Dabei durchläuft es allerdings andauernd die WHILE-Schleife. Das ist eine sehr uneffiziente Nutzung des Computers, der in dieser Wartezeit eigentlich schon wieder andere Aufgaben erledigen könnte. Effizienter wäre es, wenn das Programm ständig irgendwelche wichtigen (und langwierigen) Aufgaben erledigen könnte und sich nur bei Bedarf um die Maus kümmern würde. Dieser »Bedarf« läßt sich relativ leicht daran feststellen, daß die Maustaste gedrückt wird. Es wäre nun aber sehr lästig für ein Programm, wenn es andauernd überprüfen müßte, ob gerade die Maustaste gedrückt ist, während es sich eigentlich um andere Dinge kümmert.

16.6.2 Die ON MOUSE GOSUB-Anweisung

Um solche pausenlosen Abfragen zu vermeiden, wurde die ON...GOSUB-Anweisung geschaffen. Im folgenden Programm kommt zum Beispiel eine Zeile vor, die *ON MOUSE GOSUB mousehandler* lautet. Diese Anweisung bedeutet, daß das Programm immer, wenn die linke Maustaste gedrückt wird, automatisch zur Subroutine *mousehandler* verzweigen soll. Dabei spielt es keine Rolle, was dieses Programm in dem Moment tut, wenn die Maustaste gedrückt wird. Amiga-BASIC sichert dann den Zustand des Programms in einem dafür vorgesehenen Puffer und springt zu dieser Subroutine. Ist diese beendet, geht es im Programm

automatisch dort wieder weiter, wo es unterbrochen wurde. Eine solche ON MOUSE GOSUB-Anweisung muß allerdings noch durch eine MOUSE ON-Anweisung »aktiviert« werden, sonst tut sie gar nichts. Genauso kann man die ON MOUSE GOSUB-Anweisung auch später wieder deaktivieren, indem man die Anweisung MOUSE OFF ausführt.

Nun aber genug der Vorrede. Das folgende Programm demonstriert eine sehr simple Anwendung der ON MOUSE GOSUB-Anweisung. Im Hauptprogramm wird dazu in einer Schleife immer wieder derselbe Text ausgegeben. Sobald Sie aber auf die Maustaste drücken, verzweigt das Programm dank der ON MOUSE GOSUB-Anweisung zur Subroutine *mousehandler*.

```
ON MOUSE GOSUB mousehandler
MOUSE ON
WHILE INKEY$=""
  PRINT "Hallo Ihr da!"
WEND
END
mousehandler:
  PRINT "Und das war ein Mausklick <<<<<——"
RETURN
```

Probieren Sie das Programm am besten einmal aus und schalten Sie dabei die *Trace*-Anzeige mit dem Befehl *Trace On* ein. Sie können dann gut verfolgen, wie immer dann, wenn Sie die Maustaste drücken, nach *mousehandler* verzweigt wird. Beachten Sie bitte, daß Sie auch dieses Programm wieder mit dem Betätigen einer beliebigen Taste verlassen können. Hinter WEND stößt das Programm dann auf die END-Anweisung, die das Programm stoppt (sonst würde es fälschlicherweise in die Subroutine *mousehandler* hineinlaufen).

Das folgende Programm zeigt schon eine etwas komplexere Anwendung der ON MOUSE GOSUB-Anweisung, die demonstriert, was Sie unter Umständen beachten müssen, wenn Sie auf einen Mausklick nicht nur mit einer simplen Meldung reagieren wollen.

```
WINDOW 2, "Maler", (40,40)-(500,180)
WINDOW OUTPUT 1
ON MOUSE GOSUB mousehandler
MOUSE ON
WHILE INKEY$=""
  PRINT "Hallo Ihr da!"
WEND
END
```

```
mousehandler:
  WINDOW OUTPUT 2
  WHILE MOUSE(0) <> 0
    PSET(MOUSE(1),MOUSE(2)),1
  WEND
  WINDOW OUTPUT 1
RETURN
```

Zu Anfang dieses Programms wird ein neues Fenster namens »Maler« geöffnet. Wegen der Anweisung WINDOW OUTPUT 1 gehen die Ausgaben in der folgenden WHILE-Schleife aber nicht in dieses, sondern in das normale Output-Fenster von Amiga-BASIC. Wie schon im ersten Programm zu diesem Thema, wird bei einem Mausklick wieder nach *mousehandler* verzweigt. Diesmal verbirgt sich aber hinter dieser Subroutine das schon bekannte kleine Mal-Programm. Solange die Maustaste gedrückt ist, stoppt deshalb die Ausgabe der Texte und Sie können im Fenster »Maler« malen. Dazu muß *mousehandler* aber zunächst dieses Fenster mit *WINDOW OUTPUT 2* zum Output-Fenster machen. Vor Abschluß der Subroutine, wenn die Maustaste also losgelassen wurde, wird der alte Zustand mit *WINDOW OUTPUT 1* wiederhergestellt.

Dieses Programm führt schon recht deutlich die Möglichkeiten (und Probleme) der ON MOUSE GOSUB-Anweisung vor. Hier laufen eigentlich zwei Programme in zwei Fenstern nebeneinander her. Das eine läuft nach dem Programmstart zunächst einmal an, wird aber bei einem Mausklick sofort von dem anderen abgelöst. Auf ähnliche Weise können Programme auch auf andere Ereignisse reagieren. Hierfür sorgen Anweisungen wie zum Beispiel ON MENU, ON TIMER und ON COLLISION, die auf Menüauswahlen, auf den Ablauf einer gewissen Zeitspanne und auf den Zusammenstoß zweier Sprites reagieren.

16.6.3 Wie Sie eigene Menüs in BASIC-Programmen verwenden

Die ON MENU-Anweisung können Sie nun auch sofort ausprobieren. Sie dient der Reaktion auf eine Menüauswahl, die vom Anwender des Programms vorgenommen wurde. Menüs werden von BASIC-Programmen fast genauso wie Mausklicks gehandhabt. Hauptunterschied zur Mausbehandlung ist, daß Menüs zunächst einmal erzeugt werden müssen. Hierzu dient die MENU-Anweisung. Sie hat drei Parameter: die Nummer des zu erzeugenden Menüs, die Nummer des zu erzeugenden Menübefehls, eine Zahl (0 oder 1), die besagt, ob dieser Menübefehl wählbar sein soll und den Text des Menübefehls selbst. Ist die Nummer des Menübefehls gleich Null, ist damit der Menütitel gemeint, es wird also ein neues Menü erzeugt.

Sie können diese Anweisung aber nicht nur zum Erzeugen von Menüs und Menübefehlen verwenden. Rufen Sie MENU mit Parametern auf, die ein schon existierendes Menü oder einen Menübefehl betreffen, so können Sie dieses Menü beziehungsweise diesen Menübefehl ändern. Menübefehle können so nachträglich ihren Text ändern, auswählbar oder unauswählbar werden und so weiter.

Das folgende Programm zeigt eine sehr triviale Anwendung eigener Menüs. Es erzeugt zwei Menüs und ermöglicht die Auswahl aus diesen mit der ON MENÜ GOSUB-Anweisung. Jede

Menüauswahl wird durch Ausgabe eines entsprechenden Textes bestätigt. Der Befehl *Beenden* im Menü *Project* schließlich tut genau das, was sein Name sagt: er stoppt das Programm.

```
MENU 1, 0, 1, "Project  "
MENU 1, 2, 1, "Beenden  "
MENU 2, 0, 1, "Getränke "
MENU 2, 1, 1, "Cola "
MENU 2, 2, 1, "Aqua Rülps "
MENU 2, 3, 1, "O-Saft "
MENU 2, 4, 0, "-----"
MENU 2, 5, 1, "Milch "
ON MENU GOSUB MenuHandler
MENU ON

ende = 0
WHILE ende = 0
WEND

MENU RESET
END

MenuHandler:
dasMenu     =MENU(0)
der Befehl  =MENU(1)

IF     dasMenu = 1 THEN
  IF derBefehl = 1 THEN
    ende = 1
  END IF
ELSEIF dasMenu = 2 THEN
  IF     derBefehl = 1 THEN
    PRINT "Cola"
  ELSEIF derBefehl = 2 THEN
    PRINT "Aqua Rülps"
  ELSEIF derBefehl = 3 THEN
    PRINT "O-Saft"
  ELSEIF derBefehl = 5 THEN
    PRINT "Milch"
  END IF
END IF
RETURN
```

Am Anfang dieses Programms werden die Menüs zunächst aufgebaut, dann werden sämtliche Menüauswahlen mit *ON MENU...* und *MENU ON* an die Subroutine *MenuHandler* geschickt.

Das Programm geht dann in eine Leerschleife, die erst dann verlassen wird, wenn die Variable *end* ungleich Null ist. Sobald das der Fall ist, wird die normale Menüleiste von Amiga-BASIC mit MENU RESET wiederhergestellt und das Programm mit END gestoppt. Falls es Ihnen einmal passieren sollte, daß das Programm (vielleicht wegen eines Tippfehlers) abbricht, können Sie die Anweisung MENU RESET auch direkt im Output-Fenster eingeben. Diese Anweisung ist nötig, damit die neu erzeugten Menüs verschwinden und die BASIC-Menüs wieder auftauchen.

Nach der bisherigen Beschreibung tut das Programm in der Leerschleife (*WHILE ende = 0 ... WEND*) scheinbar gar nichts. Das täuscht aber, denn immer, wenn Sie einen Menübefehl auswählen, wird die Subroutine *MenuHandler* aufgerufen. Diese stellt dann zunächst fest, welcher Befehl in welchem Menü ausgewählt wurde. Dazu wird die Funktion MENU (nicht zu verwechseln mit der Anweisung MENU) aufgerufen. MENU(0) ergibt die Nummer des gewählten Menüs und MENU(1) die Nummer des gewählten Befehls in diesem Menü. Diese Nummern sind dabei dieselben Nummern, die bei der Erzeugung dieses Menüs (mit der MENU-Anweisung) verwendet wurden. Mit den beiden Nummern, die in den Variablen *derBefehl* und *dasMenu* gespeichert werden, geht es dann in eine große Fallunterscheidung hinein. Diese Fallunterscheidung besteht aus einer Reihe von ineinander verschachtelten IF-ELSEIF-ENDIF-Anweisungen. Sie lernen damit auch gleich die interessante Möglichkeit kennen, eine IF-Anweisung auf mehrere Zeilen zu verteilen, die auch nicht jedes BASIC zu bieten hat. Ich habe versucht, die Schachtelung durch Einrücken der Zeilen so deutlich zu machen, daß Sie eigentlich sofort verstehen müßten, wie eine solche verschachtelte IF-ELSEIF-ENDIF-Anweisung funktioniert.

Die äußere IF-Anweisung unterscheidet nur zwei Fälle, da es nur zwei Menüs gibt. Sie endet – wie jede mehrzeilige IF-Anweisung – in der letzten Programmzeile mit einem END IF. Sie besitzt zwei »Zweige«, die den beiden Menüs entsprechen, und jeder dieser Zweige enthält wieder eine IF-ELSEIF-ENDIF-Anweisung, die nach den einzelnen Befehlen im jeweiligen Menü unterscheidet. Zu jedem Menübefehl wird dann eine PRINT-Anweisung aufgerufen, die meldet, welche Wahl getroffen wurde. Nur, wenn *derBefehl* und *dasMenu* beide gleich 1 sind (der Befehl *Beenden* wurde gewählt) wird die Variable *ende* auf 1 gesetzt. Nach Beendigung der Subroutine *MenuHandler* trifft die Bedingung der WHILE-Schleife dann nicht mehr zu und das Programm wird beendet.

So, das war eigentlich schon alles, was Sie wissen müssen, um in Ihren eigenen Programmen Menüs zu verwenden. Vergessen Sie nur hinter ON MENU GOSUB nie die MENU ON-Anweisung (sonst funktioniert ON MENU GOSUB nicht) und gewöhnen Sie sich es an, am Ende Ihrer Programme immer mit MENU RESET die normale BASIC-Menüleiste wiederherzustellen!

16.7 Klangerzeugung mit Amiga-BASIC

Nachdem Sie die grafischen Möglichkeiten von Amiga-BASIC jetzt schon (andeutungsweise) kennengelernt haben, möchte ich Ihnen aber auch die wichtigsten BASIC-Befehle vorstellen,

die Sie zur Klangerzeugung benötigen. Zunächst geht es dabei in diesem Abschnitt um die Erzeugung beliebiger »Töne« und dann um ganz spezielle Töne: die synthetische Sprache.

16.7.1 Ein paar schräge Töne

Die Klangerzeugung mit Amiga-BASIC kreist um einen einzigen Befehl: SOUND. Mit diesem Befehl können Sie einen Ton bestimmter Tonhöhe, Lautstärke und Dauer aus einem der vier Tonkanäle, über die der Amiga 500 verfügt, ausgeben lassen. Dazu benötigt SOUND vier Parameter, *Frequenz*, *Dauer*, *Lautstärke* und *Kanal*. Die Tonhöhe (Frequenz) wird in Hz (Hertz) gemessen und die Dauer des Klangs in Takten (wobei 18,2 Takte laut Handbuch eine Sekunde dauern). Die Frequenz darf zwischen 20 und 15000 liegen und die Dauer zwischen 0 und 77. Die *Lautstärke* muß als Zahl zwischen 0 (stumm) und 255 (maximale Lautstärke) angegeben werden, und als *Kanal* stehen die Zahlen 0 bis 3 zur Verfügung. *Lautstärke* und *Kanal* sind dabei optional, das heißt, Sie können sie fortfallen lassen und Amiga-BASIC verwendet dann Standardwerte (maximale Lautstärke und Kanal Nummer 0).

Geben Sie testweise einmal die folgende SOUND-Anweisung im Output-Fenster ein und drücken Sie die [Return]-Taste:

```
SOUND 440, 10
```

Gleichzeitig mit der *Ok*-Meldung müßte dann ein Ton erklingen (es ist der Kammerton A mit einer Frequenz von 440 Hz). Damit Sie etwas hören, müssen Sie allerdings die beiden Sound-Ausgänge an der Rückseite des Amiga 500 mit einem Paar Lautsprecher oder über ein spezielles Kabel mit Ihrem Monitor verbunden haben.

Wenn Sie Lust haben, können Sie nun auch einmal andere Werte für die Parameter *Frequenz* und *Tondauer* ausprobieren. Achten Sie dabei auch darauf, daß der einmal »angestoßene« Ton auch dann noch »nachklingt«, wenn die *Ok*-Meldung bereits erschienen ist und Sie schon den nächsten Befehl eingeben können. (Am auffälligsten ist das natürlich, wenn die Tondauer länger wird.) Dies läßt eine erste Ahnung davon aufkommen, daß die Hardware (und Systemsoftware) einen Ton selbständig erzeugt und das BASIC-Programm dazu nicht mehr benötigt wird, wenn es einmal den Ton angestoßen hat. Um das noch etwas deutlicher zu zeigen, geben Sie bitte einmal das folgende Programm ein:

```
PRINT "Start !"
FOR i = 1 TO 10
  SOUND 4000, 10, 255
  SOUND 200, 10, 128
NEXT i
PRINT "Fertig !"
```

Dieses Programm läßt zwanzig Töne erklingen. Ein hoher leiserer und ein tieferer lauter Ton wechseln sich dabei immer ab. (Wie gut sie klingen, hängt ein bißchen von dem Lautsprecher oder Monitor ab, den Sie angeschlossen haben.) Erstaunlich an diesem Programm ist aber vor allem, daß ein Großteil der Töne erst erklingt, wenn das Programm längst beendet ist. Es ist nämlich so, daß nicht einmal zum »Anstoßen« eines Tons ein Programm laufen muß. Jeder

SOUND-Befehl wird zunächst einmal in einer Warteschlange gespeichert. Für jeden der vier Tonkanäle gibt es eine eigene Warteschlange für diesen Zweck. Sobald ein Tonkanal nichts mehr zu tun hat (also der vorige Ton verklungen ist) überprüft die Systemsoftware des Amiga, ob die zugehörige Warteschlange leer ist. Ist sie das nicht, wird der älteste SOUND-Befehl, der schon am längsten wartet, aus der Warteschlange genommen und der entsprechende Ton erklingt. Ist dieser verklungen, ist der nächste auf diesen Kanal wartende SOUND-Befehl an der Reihe. Das alles läuft völlig unabhängig davon ab, was das laufende BASIC-Programm gerade tut. Es muß – wie oben demonstriert – noch nicht einmal ein aktives BASIC-Programm geben.

Das folgende Programm demonstriert diese Eigenschaft der Sounderzeugung auf dem Amiga noch eindrucksvoller. Es benutzt dazu die Befehle SOUND WAIT, der verhindert, daß SOUND-Befehle aus einer Warteschlange genommen werden, und SOUND RESUME, der die Soundmaschinerie wieder in Gang setzt.

```
SOUND WAIT
PRINT "Start !"
FOR i = 1 TO 5
  SOUND 4000, 10, 255
  SOUND 200, 10, 128
NEXT i
SOUND RESUME
PRINT "Fertig !"
```

Die vier Warteschlangen können leider aber nicht beliebig lang werden. Sie können also nicht SOUND WAIT aufrufen, ein beliebig langes Musikstück in die vier Warteschlangen legen und es danach mit SOUND RESUME erklingen lassen, während Sie längst an einem ganz anderen Programm arbeiten. Falls die Warteschlange überläuft, erscheint ein Alert, der Ihnen meldet, daß kein Speicher mehr für diesen Zweck da ist (*Out of heap space. Press left mouse button, to continue*). Um diesen Fehler zu vermeiden, wurde übrigens in dem obigen Programm die obere Grenze der FOR-Schleife auf 5 heruntergesetzt.

Apropos Musik: Ich habe Ihnen in diesem Kapitel ja nur gezeigt, wie man recht schräge Töne erzeugen kann. Zu mehr reicht mein musikalisches Talent nicht aus. Wenn Sie einmal sehen wollen, wie man richtige Musik mit dem Amiga erzeugen kann, schauen Sie sich doch einfach einmal das Beispielprogramm *Music* in der Schublade *BasicDemos* auf der Extras-Diskette an. Es zeigt auf eindrucksvolle Weise, wie man ein Programm eine komplizierte mehrstimmige Melodie spielen lassen kann und dabei gleichzeitig noch eine hübsche Grafik auf den Bildschirm bringt.

16.7.2 Sprachsynthese

Nun möchte ich Ihnen aber auch noch den zweiten wichtigen Aspekt der Sounderzeugung auf dem Amiga 500 vorstellen, die Erzeugung synthetischer Sprache (Sprachsynthese). Dies ist von Amiga-BASIC aus so lächerlich einfach, daß ich mich fast schäme, das folgende Programm vorzustellen. Es kann fast genausoviel wie das Werkzeug *Say*, das in Kapitel 5

vorgestellt wurde. Um dieses Programm auszuprobieren, dürfen Sie aber auf keinen Fall die Sprachbibliotheken von Ihrer Workbench-Diskette entfernt haben, wie es im Kapitel 11 beschrieben wurde!

```
Sag$ = "X"
WHILE Sag$<>""
  LINE INPUT "Was soll ich sagen? ";Sag$
  Phonem$ = TRANSLATE$(Sag$)
  PRINT "Phoneme  : ";Phonem$
  SAY(Phonem$)
WEND
```

Das Programm ist in eine große WHILE-Schleife eingebettet, die abbricht, sobald die Variable *Sag$* leer ist. (Deshalb wird *Sag$* zu Anfang auf »X« gesetzt). In der WHILE-Schleife wird jeweils eine ganze Zeile (bis [Return]) über die Tastatur eingelesen und in *Sag$* gespeichert. Dieser Text wird dann mit der Funktion TRANSLATE$ in Phoneme umgewandelt. Und eine Liste von Phonemen kann man mit SAY aussprechen lassen – was das Programm dann auch tut. (Der Zusammenhang zwischen Sprache und Phonemen ist bereits in Kapitel 13 erläutert worden.) Das klingt wahrscheinlich nicht besonders schön. Wie Sie ja wissen, versucht der Amiga jeden Text zunächst einmal auszusprechen, als wären es englische Wörter.

Sie können das nur umgehen, indem Sie die SAY-Funktion direkt ohne Umweg über TRANSLATE verwenden und die Phoneme, die ausgesprochen werden sollen, selbst (zum Beispiel anhand eines Wörterbuches) ermitteln. Eine andere Lösung ist es, selbst ein Programm zu entwickeln, das deutsche Sprache in die richtigen Phoneme umwandelt. Das ist allerdings nicht ganz so einfach. Glücklicherweise hat Ihnen aber schon jemand diese Arbeit abgenommen. Auf der Extras-Diskette befindet sich in der Schublade *BasicDemos* ein Programm namens *SpeechD*, das unter anderem auch deutsche Texte (einigermaßen) gut aussprechen kann. Dieses Programm kann auch sonst einiges mehr als das obige Mini-Sprachprogramm. Es erlaubt Ihnen unter anderem auch, Sprechgeschwindigkeit und Tonhöhe einzustellen. Wenn Sie also wissen wollen, wie man so etwas in einem BASIC-Programm macht, schauen Sie sich *SpeechD* ruhig einmal im Listing-Fenster an.

16.8 Libraries: Amiga-BASIC ohne Grenzen

Obwohl Amiga-BASIC bereits eine Vielzahl von Befehlen enthält, kann es doch immer wieder vorkommen, daß zusätzliche Fähigkeiten benötigt werden, die in BASIC nicht realisierbar sind. Von BASIC aus können zum Beispiel keine Requester und Gadgets erzeugt und benutzt werden. Fortgeschrittene Programmierer haben aber dennoch die Möglichkeit, ohne allzu große Mühe auch diese Einschränkungen zu umgehen. Diese Möglichkeiten sind aber »mit Vorsicht zu genießen«, da einige Schritte dabei recht fehleranfällig sind.

Funktionen, die diese fehlenden Fähigkeiten realisieren, befinden sich nämlich zum großen Teil bereits in den Bibliotheks-Dateien, aus denen die Systemsoftware des Amiga zum Teil besteht.

Amiga-BASIC besitzt eine recht elegante Konstruktion, mit der es möglich ist, solche Funktionen auch in einem BASIC-Programm zu verwenden, als wären sie in BASIC geschrieben und würden sich im selben Programm befinden. Mit dem LIBRARY-Befehl können eine Reihe solcher Bibliotheks-Dateien angegeben werden, die Amiga-BASIC durchsucht, um diese Befehle zu finden. Wird danach in einem Programm ein Befehl verwendet, der nicht im Programm selbst als Subprogramm definiert ist, versucht Amiga-BASIC, ihn in einer Bibliothek (engl. *library*) zu finden.

Solche Bibliotheken müssen aber nicht zur Amiga-Systemsoftware gehören. Auch eigene Funktionspakete, die man zum Beispiel aus Geschwindigkeitsgründen in Maschinensprache geschrieben hat, würde man typischerweise in Bibliotheken ablegen. Für BASIC spielt dies keine Rolle. Die einzelnen Funktionen müssen sich nur an gewisse Aufruf-Konventionen halten, die zusammen mit den Funktionsnamen in einer von der eigentlichen Bibliothek getrennten Datei verzeichnet sein müssen. Diese Datei trägt immer denselben Namen wie die Bibliotheks-Datei selbst, aber mit einem zusätzlichen ».bmap« dahinter.

Die Beschreibungen der Bibliotheksfunktionen kann man zunächst in recht lesbarer Form erstellen. Das Format dafür ist im Anhang des Amiga-BASIC-Handbuchs erläutert. Es handelt sich dabei um die sogenannten FD-Dateien. Ein besonderes Programm namens *ConvertFd* auf der Extras-Diskette wandelt diese Beschreibung aus der FD-Datei dann in das kompakte ».bmap»-Format um, das der LIBRARY-Befehl verlangt. Zu den wichtigsten Bibliotheken befinden sich aber schon bmap-Dateien in der Schublade *BasicDemos* auf der Extras-Diskette. Sie werden also wahrscheinlich nie in die Verlegenheit kommen, ConvertFd.bas zu brauchen. Falls doch, finden Sie alle nötigen FD-Dateien in der Schublade *FD1.2* auf der Extras-Diskette.

Um die in den Systembibliotheken versammelten Funktionen aber sinnvoll nutzen zu können, benötigen Sie weit mehr Wissen über die Programmierung des Amiga, als ich Ihnen in diesem Kapitel verschaffen konnte. Ich erspare mir deshalb ein Beispielprogramm, das LIBRARY-Funktionen verwendet.

Und das war es dann auch schon! Ich hoffe, Sie haben in diesem Kapitel gesehen, daß Amiga-BASIC nicht nur ein vielseitiger und komfortabler Dialekt der Programmiersprache BASIC, sondern auch grenzenlos erweiterbar ist. Auch ohne diese Erweiterungen sind aber schon mit wenigen Anweisungen interessante Programme möglich, wie Sie anhand der oben vorgestellten Beispielprogramme sehen konnten. Amiga-BASIC ist deshalb vielleicht die einzige Sprache, die Sie jemals benötigen werden. Fairerweise muß man jedoch dazu sagen, daß es viele Programmierprobleme gibt, für die man vielleicht doch besser eine andere Programmiersprache einsetzen sollte – aber eben nicht unbedingt muß.

Anhang: Fehlercodes und ihre Bedeutung

Sowohl im CLI wie auch auf der Workbench werden Ihnen gelegentlich Fehlermeldungen gezeigt. Bei CLI-Kommandos werden diese Meldungen im CLI-Fenster ausgegeben. Andere Programme zeigen Ihnen dazu vielleicht einen Requester. Die Workbench beschränkt sich oft auf einen Piepston und eine knappe Meldung in der Titelleiste des Workbench-Screens. Diese Meldung können Sie sich übrigens noch einmal anzeigen lassen, indem Sie den Befehl *Last Error* aufrufen. Alle diese Fehlermeldungen haben aber eine gemeinsame Eigenschaft: sie sind recht knapp und oft unverständlich. Deshalb folgt an dieser Stelle nun eine etwas ausführlichere Beschreibung der Fehler, die bei der Bedienung des Amiga 500 auftreten können.

Jede dieser Fehlerbeschreibungen beginnt mit einer Nummer und einer knappen englischen Meldung. Dies sind die beiden Komponenten einer Fehlermeldung, die Sie auch sonst geboten bekommen. (Der Fehlercode – die Nummer – dient zur eindeutigen Identifizierung des Fehlers.) Dann folgt eine kurze Übersetzung der Fehlermeldung und eine etwas längere Erläuterung der möglichen Fehlerursache. Zu jedem Fehler finden Sie dann auch noch einen oder mehrere Vorschläge, wie Sie das Problem eventuell beheben könnten.

Diese Liste ist übrigens nicht vollständig. Unter gewissen Umständen können auch Fehler auftreten, die hier nicht verzeichnet sind. In diesen Fällen treten die Fehler aber durch interne Fehler in Programmen auf, auf die Sie keinen Einfluß haben. Eine Erläuterung solcher Fehlermeldungen würde Ihnen also wenig helfen!

103: *insufficient free store*

Freier RAM-Speicher reicht nicht aus.

Erläuterung: Sie haben versucht, etwas zu tun, wofür RAM-Speicher benötigt wird – zum Beispiel ein Programm gestartet. Es ist aber nicht mehr genügend Speicher frei.

Gegenmaßnahmen: Sie müssen versuchen Speicher freizumachen. Beenden Sie dazu zunächst alle Programme, die Sie im Augenblick nicht unbedingt benötigen und schließen Sie möglichst viele Fenster. Wenn das immer noch nicht hilft, versuchen Sie es nach einem Neustart noch einmal. Wenn das auch nichts hilft, sollten Sie die Anschaffung einer Speichererweiterung ins Auge fassen.

120: *argument line invalid or too long*

Kommandozeile ist zu lang oder enthält fehlerhafte Teile.

Erläuterung: Sie haben versucht, einen CLI-Befehl einzugeben, der zu lang ist oder ein falsches Format für die Parameter verwendet.

Gegenmaßnahmen: Es sollte eigentlich nie nötig sein, einen Befehl einzugeben, der mit allen Parametern länger wird als eine Zeile. Falls Sie sehr lange Pfadnamen als Parameter verwenden müssen, sollten Sie das aktuelle Directory wechseln.

121: *file is not an object module*

Datei ist keine Programmdatei.

Erläuterung: Sie haben versucht, eine Datei wie ein Werkzeug oder CLI-Kommando zu starten, aber diese Datei hat nicht das korrekte Format einer Programmdatei.

Gegenmaßnahmen: Prüfen Sie – falls dieser Fehler nach einem CLI-Befehl auftaucht – ob Sie sich nicht verschrieben haben. Schlagen Sie im Zweifelsfall den Befehl im Kapitel 8 nach. Beachten Sie bitte auch, daß Sie Kommandofolgen mit *execute* aufrufen müssen. Auf der Workbench kann dieser Fehler eigentlich nur dann vorkommen, wenn Sie Info-Dateien mit dem *IconEd* oder im CLI manipuliert haben und dabei einer Datei ein Werkzeug-Piktogramm verschafft haben, obwohl die Datei kein Werkzeug ist. Falls Sie alle diese Ursachen ausschließen können, kann es auch noch sein, daß die Programmdatei defekt ist. In diesem Fall sollten Sie sie löschen und durch eine Sicherungskopie ersetzen.

202: *object in use*

Objekt in Gebrauch.

Erläuterung: Sie – oder ein von Ihnen benutztes Programm – haben versucht, eine Datei zu manipulieren, die von einem anderen Programm benötigt wird.

Gegenmaßnahmen: Eine häufige Ursache für diesen Fehler ist der Versuch, eine Datei oder ein Directory zu löschen, die (das) in Gebrauch ist. Ein Beispiel hierfür wäre eine Kommandofolge, die versucht, sich mit *delete* selbst zu löschen. Das geht natürlich nicht. Wollen Sie diese Datei wirklich löschen, sollten Sie dieses andere Programm beenden und es dann noch einmal versuchen.

203: object already exists

Eine Datei oder ein Directory dieses Namens existieren schon.

Erläuterung: Sie können beim Umbenennen eines Piktogramms auf der Workbench oder einer Datei im CLI nicht den Namen einer schon existierenden Datei oder eines schon existierenden Directory verwenden. Auf der Workbench kann es sogar vorkommen, daß ein Umbenennen auch dann nicht möglich ist, wenn in derselben Schublade kein Piktogramm gleichen Namens existiert. In diesem Fall gibt es aber vielleicht eine unsichtbare Datei (ohne Piktogramm), die diesen Namenskonflikt verursacht.

Gegenmaßnahmen: Wenn eine umbenannte Datei die alte Datei gleichen Namens ersetzen soll, müssen Sie sie vorher löschen. Ansonsten müssen Sie einen anderen Namen verwenden.

204: directory not found

Directory nicht vorhanden.

Erläuterung: In einem Namen oder Pfadnamen, den Sie eingegeben haben, kommt ein Directory vor, das es nicht gibt. Es kann auch sein, daß Sie einen Doppelklick auf ein Schubladen-Piktogramm gemacht haben, zu dem es kein Directory gibt. Das kann leicht vorkommen, wenn Sie mit dem Piktogramm-Editor *IconEd* herumspielen.

Gegenmaßnahmen: Kontrollieren Sie die Schreibweise des eventuell eingegebenen Pfadnamens. In langen Pfadnamen verschreibt man sich leicht. Tritt diese Meldung beim Öffnen eines Schubladen-Piktogramms auf, sollten Sie diese Schublade mit *Discard* löschen (sie ist sowieso leer).

205: object not found

Datei nicht vorhanden

Erläuterung: Entweder Sie haben den Namen eines Programms eingegeben und Amiga-DOS kann diese Datei nicht finden, oder Sie haben versucht, einen Doppelklick auf ein Projekt-Piktogramm zu machen und die Workbench kann das zugehörige Werkzeug nicht finden.

Gegenmaßnahmen: Wenn dieser Fehler im CLI auftritt, sollten Sie wie üblich die Schreibweise des Befehls und aller Parameter im letzten Kommando überprüfen. Besonders in langen Pfadnamen verschreibt man sich leicht. Passiert dieser Fehler jedoch auf der Workbench, so befindet sich das Werkzeug zu einem Projekt nicht dort, wo es die Workbench erwartet. Öffnen Sie dann (mit dem Befehl *Info*) das Info-Fenster dieses Projekts und tragen Sie dort den vollen Pfadnamen des zugehörigen Werkzeugs ein. Falls die Workbench das Werkzeug nur deshalb nicht finden konnte, weil die entsprechende Diskette nicht eingelegt war, kann es auch reichen, die Diskette in ein Laufwerk einzulegen und das Projekt noch einmal zu öffnen.

210: invalid stream component name

Ein Teil eines (Pfad-) Namens ist nicht korrekt.

Erläuterung: Sie haben – zum Beispiel beim Umbenennen oder Kopieren einer Datei – einen Namen eingegeben, der nicht zulässig ist.

Gegenmaßnahmen: Verwenden Sie in Namen, die Sie Dateien, Directories und Piktogrammen geben, möglichst keine Sonderzeichen. Vor allem auf »:« und »/« sollten Sie verzichten. Zudem dürfen solche Namen maximal 30 Zeichen lang sein. Pfadnamen können natürlich länger werden – nicht aber die einzelnen Komponenten eines Pfadnamens zwischen den Schrägstrichen und Doppelpunkten.

212: object not of required type

Objekt besitzt nicht den richtigen Typ.

Erläuterung: Sie haben bei einem CLI-Kommando den Namen eines Directory angegeben, wo ein Dateiname erwartet wurde, oder umgekehrt.

Gegenmaßnahmen: Schlagen Sie in Kapitel 8 die genaue Funktion des jeweiligen Befehls nach. Sie können als Argument von *type* zum Beispiel kein Directory angeben. Vielleicht haben Sie sich aber auch nur verschrieben oder eine Komponente eines langen Pfadnamens vergessen.

213: disk not validated

Diskette konnte nicht validiert werden.

Erläuterung: Amiga-DOS versucht nach jedem Einlegen einer Diskette, diese zu »validieren«. Dabei wird überprüft, ob die internen Datenstrukturen, die Amiga-DOS benötigt, um die Daten auf der Diskette wiederzufinden, vorhanden und in Ordnung sind. Dieser Vorgang ist langwierig und deshalb kann die entsprechende Fehlermeldung eine ganze Weile nach dem Einlegen der Diskette auftauchen. Solange eine Diskette nicht validiert ist, können Sie übrigens auch keine neuen Dateien darauf anlegen (zum Beispiel durch Kopieren einer anderen Datei).

Gegenmaßnahmen: Wenn Sie diese Fehlermeldung erhalten, sieht es recht schlecht für die jeweilige Diskette aus. Machen Sie eine Kopie davon und versuchen Sie diese Kopie mit *diskdoctor* (siehe Kapitel 8) wiederherzustellen. Kopieren Sie dann alle intakten Dateien auf eine neue leere Diskette und formatieren Sie die defekte Diskette neu.

214: disk write-protected

Diskette ist schreibgeschützt.

Erläuterung: Sie haben versucht, eine Schreiboperation mit einer Diskette durchzuführen, bei der der Schreibschutz aktiviert ist. Beachten Sie bitte, daß Schreiboperationen oft auch dann nötig sind, wenn Sie scheinbar keine entsprechende Aktion durchgeführt haben. Wenn Sie zum

Beispiel im Hauptfenster von *Preferences* auf SAVE klicken oder einen Schnappschuß machen, müssen Daten auf die Diskette geschrieben werden.

Gegenmaßnahmen: Nehmen Sie die Diskette aus dem Laufwerk, entfernen Sie den Schreibschutz und wiederholen Sie die Aktion, die zu dem Fehler führte.

215: *rename across devices attempted*

Umbenennung von einem Gerät zum anderen versucht.

Erläuterung: Sie haben versucht, eine Datei »auf eine andere Diskette umzubenennen«. Das geht nicht. Der neue Name einer Datei kann zwar in einem anderen Directory liegen, aber nicht auf einer anderen Diskette.

Gegenmaßnahmen: Kopieren Sie zuerst die Datei auf die andere Diskette und löschen Sie dann das Original.

216: *directory not empty*

Directory ist nicht leer.

Erläuterung: Sie haben im CLI versucht, ein Directory, das noch andere Directories oder Dateien enthält, mit dem *delete*-Befehl zu löschen. Directories können aber nur dann gelöscht werden, wenn sie leer sind.

Gegenmaßnahmen: Leeren Sie das Directory, bevor Sie es löschen oder verwenden Sie die *ALL*-Option des *delete*-Befehls (siehe Kapitel 8).

218: *device not mounted*

Gerät oder Diskette stehen im Moment nicht zur Verfügung.

Erläuterung: Ein Programm hat versucht, eine Diskette anzusprechen, die im Moment in keinem Laufwerk liegt (aber früher schon einmal eingelegt worden war).

Gegenmaßnahmen: Legen Sie die gewünschte Diskette ein und versuchen Sie dann noch einmal, die Arbeitsschritte durchzuführen, die zu dem Fehler führten.

220: *comment too big*

Kommentar ist zu lang.

Erläuterung: Amiga-DOS verwaltet zu jeder Datei einen Kommentar, in dem Sie zusätzliche Informationen unterbringen können, die nicht in den Dateinamen passen. Dieser Kommentar darf maximal 80 Zeichen lang sein und kann im Info-Fenster oder mit dem Befehl *filenote* verändert werden.

Gegenmaßnahmen: Versuchen Sie, sich kürzer zu fassen und einen Kommentar mit weniger als 80 Zeichen zu formulieren.

221: disk full

Diskette ist voll.

Erläuterung: Eine Diskette kann eine bestimmte Maximalmenge an Daten aufnehmen (circa 900 000 Zeichen oder 880 Kbyte). Die Fehlermeldung *disk full* erscheint, wenn bei einer Schreiboperation versucht wird, diese Grenze zu überschreiten. Das kann beim Kopieren einer Datei passieren oder auch beim Abspeichern einer Datei nach Änderungen.

Gegenmaßnahmen: Schaffen Sie Platz auf der Diskette. Leeren Sie dazu zuerst den Papierkorb (mit dem Workbench-Befehl *Empty Trash*) und löschen Sie dann nicht mehr benötigte Dateien und Directories (Schubladen). Wenn das alles nichts hilft, müssen Sie eine neue leere Diskette verwenden, um die Daten abzuspeichern, die die Fehlermeldung verursacht haben.

222: file is protected from deletion

Datei ist vor dem Löschen geschützt.

Erläuterung: Sie haben versucht, eine Datei zu löschen, deren Löschschutz aktiviert ist.

Gegenmaßnahmen: Wenn Sie die so geschützte Datei wirklich löschen wollen, brauchen Sie diesen Löschschutz nur zu entfernen und dann die Löschoperation zu wiederholen. Den Löschschutz einer Datei können Sie im Info-Fenster dieser Datei oder mit dem CLI-Kommando *protect* (siehe Kapitel 8) entfernen.

223: file is write protected

Datei ist vor dem Überschreiben geschützt.

Erläuterung: Sie haben versucht, eine Datei zu überschreiben, deren Überschreibungsschutz aktiviert ist.

Gegenmaßnahmen: Wenn Sie die so geschützte Datei wirklich überschreiben wollen, brauchen Sie den Überschreibungsschutz nur zu entfernen und dann die Operation zu wiederholen, die zu der Fehlermeldung führte. Den Überschreibungsschutz einer Datei können Sie mit dem CLI-Kommando *protect* (siehe Kapitel 8) entfernen.

225: not a valid DOS disk

Keine Amiga-DOS-Diskette.

Erläuterung: Es gibt viele Computer, die dieselben Disketten verwenden wie der Amiga 500 (zum Beispiel der Atari ST, der Apple Macintosh und eine Reihe tragbarer Computer). Alle diese Computer verwenden verschiedene Formate, um die Daten auf Diskette zu schreiben, die der Amiga 500 »nicht versteht«. Es kann aber auch sein, daß Sie versehentlich eine sogenannte *Kickstart-Diskette* des Amiga 1000 eingelegt haben. Auch diese Diskette können Sie auf der Workbench und im CLI nicht bearbeiten.

Gegenmaßnahmen: Sie brauchen die Diskette nur wieder aus dem Laufwerk zu nehmen, wenn dieser Befehl auftaucht. Wollen Sie auf dieser Diskette jedoch Daten abspeichern, müssen Sie sie zunächst initialisieren beziehungsweise formatieren.

226: no disk in drive

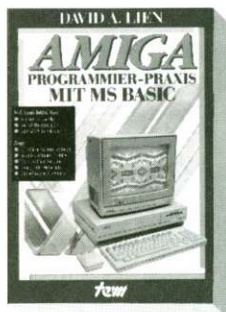
Keine Diskette im Laufwerk.

Erläuterung: Ein Programm hat versucht auf ein Laufwerk zuzugreifen (zum Beispiel beim Kopieren einer Diskette), und es befindet sich keine Diskette in diesem Laufwerk. Dieser Fehler tritt aber sehr selten auf. Die meisten Programme zeigen in solchen Fällen einen Requester, der Sie auffordert, die Diskette einzulegen.

Gegenmaßnahmen: Legen Sie die gewünschte Diskette in das Laufwerk.

Stichwortverzeichnis

Bücher zum Amiga



Prof. D. A. Lien
Amiga: Programmier-Praxis mit MS BASIC
1986, 400 Seiten
Bestseller! Eine systematische und lebendige Einführung in MS BASIC unter der komfortablen Mausbedienung und Fensteroberfläche des Amiga. Mit über 60 Musterprogrammen zu den Befehlen. Zeigt Amiga-typische Anwendungen: bewegte/farbige Grafiken; Musik- und Sprachausgabe, Strings, Felder, Mathematik, Dateibehandlung, Ein-/Ausgabe sowie »Entwurf von Programmen«.
Best.-Nr. 80369
ISBN 3-921803-69-1
DM 59,–

H.-R. Henning
Programmieren mit Amiga-BASIC
1987, ca. 360 Seiten, inkl. Diskette
Einführung in die Programmierung des Amiga-BASIC: Grafik, Sprites, Sprachausgabe, sequentielle Dateien, Fenstertechnik, Musik, Tips und Tricks.
Hard- und Software-Anforderungen: Amiga 500, 1000 oder 2000 mit 512 Kbyte Arbeitsspeicher, gegebenenfalls ein grafikfähiger Matrixdrucker und ein Joystick, Amiga-BASIC von Microsoft
Best.-Nr. 90434,
ISBN 3-89090-434-3
DM 59,–

W. Hilf/A. Nausch
M68000-Familie: Teil 1 Grundlagen und Architektur
1984, 568 Seiten
Ausbildungs- und Entwicklungstext mit allen notwendigen Informationen über den M68000.
Best.-Nr. 80316
ISBN 3-921803-16-0
DM 79,–

W. Hilf/A. Nausch
M68000-Familie: Teil 2 Anwendungen und 68000-Bausteine
1985, 400 Seiten
In vielen Programmierbeispielen liefert dieses Buch die Praxis der in Teil 1 vermittelten Theorie.
Best.-Nr. 80330
ISBN 3-921803-30-6
DM 69,–

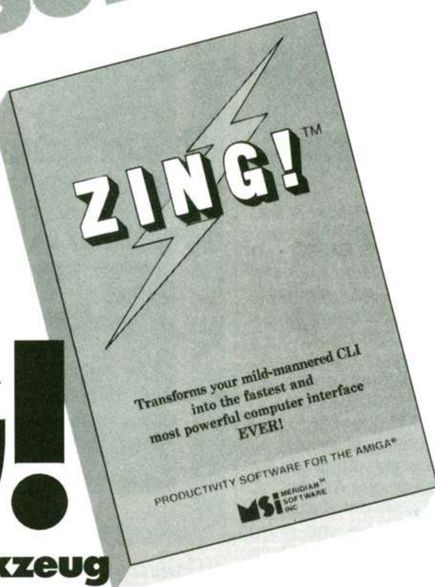
707327

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Brandneue

stop · unentbehrlich für jeden Amiga-User · stop · frisch bei Markt&Technik eingetroffen · stop · deutsche Programmversionen in Arbeit · stop · exklusiv bei Markt&Technik · stop · Update-Service für alle unsere Kunden · stop

Amiga-Software



ZING!

Das mächtige CLI-Werkzeug

Mit ZING! haben Sie endlich das gesamte File-System mit Directories und Subdirectories fest im Griff. Sie beschleunigen mit ZING! alle nötigen Arbeiten mit Files, verwalten bis zu 500 Files und Subfiles und bis zu 100 Directories auf einmal.
Die Bedieneroberfläche ist vom Feinsten:
- Pull-down-Menüs
- (Click-)Icons
- Funktionstasten

Weitere Optionen wie: Task-Monitor, Printer-Spooler, Screen-Syer/Printer, Screen-Dimmer, Veränderung der Voreinstellung der Funktionstasten und des Systems.
Am besten gleich bestellen!

Best.-Nr. 52571

DM 189,–*

*inkl. MwSt. unverbindliche Preisempfehlung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

707324

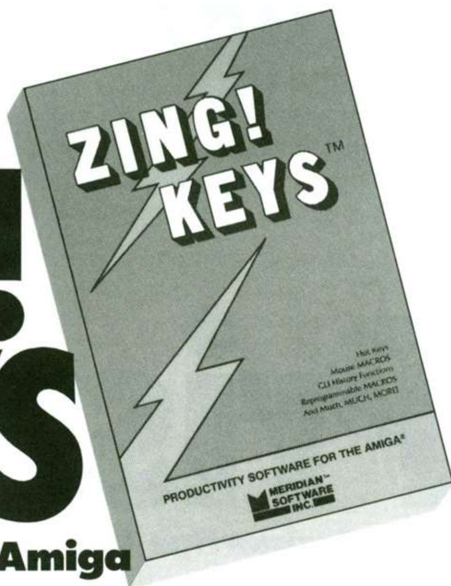
Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Brandneue

stop · unentbehrlich für jeden Amiga-User · stop · frisch bei Markt&Technik
eingetroffen · stop · deutsche Programmversionen in Arbeit · stop · exklusiv bei
Markt&Technik · stop · Update-Service für alle unsere Kunden · stop

Amiga-Software

ZING! KEYS



Ihr ganz persönlicher Amiga

Mit ZING!KEYS machen Sie aus Ihrem Amiga das variable System, das Sie sich schon immer wünschen. Es ist Ihren eigenen Ansprüchen jederzeit anpaßbar! Alle Tasten sind nach Wunsch belegbar: z.B. mit Funktionsaufrufen, Programmaufrufen, Systembefehlen und vorprogrammierten Befehlen. Die Belegung ist natürlich jederzeit abspeicherbar.

Durch die Belegung von »Hot-Keys« haben Sie mit ZING! KEYS ein Multitaskingsystem par excellence!

Best.-Nr. 52572

DM 109,–*

*inkl. MwSt. unverbindliche Preisempfehlung

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

707325

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Brandneue

stop · unentbehrlich für jeden Amiga-User · stop · frisch bei Markt&Technik eingetroffen · stop · deutsche Programmversionen in Arbeit · stop · exklusiv bei Markt&Technik · stop · Update-Service für alle unsere Kunden · stop

Amiga-Software

Prism

Das einzigartige 4096-Farben-Grafikprogramm

Prism ist nicht nur ein neues Zeichenprogramm unter vielen - Prism ist mehr! Denn Prism beschränkt sich nicht auf die übliche 32-Farben-Palette: Mit Prism haben Sie die ganze Bandbreite der 4096 Farbschattierungen des Amiga zur Verfügung. Auf einmal und in einem Bild! Und Sie können bis zu 4096 neue Farbtöne zu den bestehenden Grafiken hinzufügen und Ausschnitte von einem auf ein anderes Bild übertragen und und und…

Ein einzigartiges Programm für digitalisierte Bilder und deren Manipulation!

Best.-Nr. 52570

DM 159,–*

*inkl. MwSt. unverbindliche Preisempfehlung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

707326

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bitte schneiden Sie diesen Coupon aus, und schicken Sie ihn in einem Kuvert an:
Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar

Computerliteratur und Software vom Spezialisten

Vom Einsteigerbuch für den Heim- oder Personalcomputer-Neuling über professionelle Programmierhandbücher bis hin zum Elektronikbuch bieten wir Ihnen interessante und topaktuelle Titel für

• Apple-Computer • Atari-Computer • Commodore 64/128/16/116/Plus 4 • Schneider-Computer • IBM-PC, XT und Kompatible
sowie zu den Fachbereichen Programmiersprachen • Betriebssysteme (CP/M, MS-DOS, Unix, Z80) • Textverarbeitung • Datenbanksysteme • Tabellenkalkulation • Integrierte Software • Mikroprozessoren • Schulungen. Außerdem finden Sie professionelle Spitzen-Programme in unserem preiswerten Software-Angebot für Amiga, Atari ST, Commodore 128, 128 D, 64, 16, für Schneider-Computer und für IBM-PCs und Kompatible!
Fordern Sie mit dem nebenstehenden Coupon unser neuestes Gesamtverzeichnis und unsere Programmservice-Übersichten an, mit hilfreichen Utilities, professionellen Anwendungen oder packenden Computerspielen!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2,
8013 Haar bei München, Telefon (089) 4613-0

Adresse:

Name

Straße

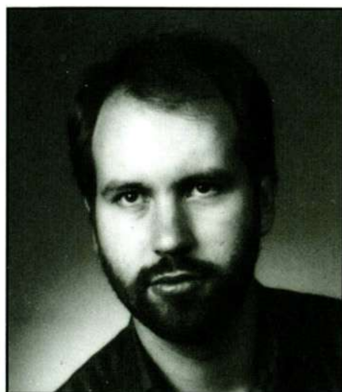
Ort

Bitte schicken Sie mir:
☐ Ihr neuestes Gesamtverzeichnis
☐ Eine Übersicht Ihres Programmservice-Angebotes aus der Zeitschrift

☐ Außerdem interessiere ich mich für folgende/n Computer:

(PS: Wir speichern Ihre Daten und verpflichten uns zur Einhaltung des Bundesdatenschutzgesetzes)

Markt&Technik Verlag AG
– Unternehmensbereich Buchverlag –
Hans-Pinsel-Straße 2
D-8013 Haar bei München



MARKUS BREUER
(Jahrgang 1960) studiert Informatik an der Universität Dortmund. Hauptinteressengebiet sind seit langem Computer mit hochentwickelten grafischen Benutzeroberflächen. Neben dem Studium ist er seit 1984 als Programmentwickler und seit 1985 als technischer Autor und EDV-Berater tätig. Verfasser einer Vielzahl von Artikeln, Büchern und Handbüchern zu den Themenkreisen Apple Macintosh, Commodore Amiga, moderne Programmiersprachen, künstliche Intelligenz und Desktop Publishing. Seit 1987 Geschäftsführer und Mitinhaber eines Systemhauses, das sich auf Textverarbeitung, Desktop Publishing und Desktop Communications (LA-Netzwerke) spezialisiert hat.

Amiga-500-Buch

Als der Commodore Amiga auf den Markt kam, glich es einer Sensation. Er spiegelte in der technologischen Entwicklung den neuesten Stand der Technik wider, von der Hardware-Architektur bis zum Multitasking-Betriebssystem. Leider lag der Preis für das erste Amiga-Modell »1000« für viele Interessenten noch außerhalb ihrer finanziellen Möglichkeiten. Der Amiga 500 bietet nun dieselbe Leistungsfähigkeit wie das erste Modell zum Preis eines einfachen Homecomputers.
Er besitzt zum Beispiel enorme Farbgrafik-Fähigkeiten (bis zu 4096 Farben gleichzeitig am Bildschirm), die auch konsequent bei der Bedienung des Betriebssystems eingesetzt werden. Darüber hinaus machen grafische Symbole (Piktogramme), ein komfortables Menüsystem und die »Maus« das Auswendiglernen komplizierter Befehlsfolgen überflüssig. Die Bedienung des Amiga 500 ist so einfach wie die Auswahl eines Getränks auf einer Speisekarte.
Das vorliegende Buch bietet zunächst eine behutsame Einführung in die Arbeit mit dem Amiga 500. Es beschränkt sich aber nicht auf diese Einführung. Ein Handbuchteil mit vielen Bildschirmfotos und Übersichtstabellen hilft Ihnen, schnell und effektiv mit dem Amiga 500 umzugehen. Daneben erhält der Leser auch eine ausführliche Beschreibung der Hardware-Architektur des Amiga 500 und seines Zubehörs.

Aus dem Inhalt:
- Erste Handgriffe auf der Workbench
- Arbeiten mit dem Amiga
- Die Kommandos des CLI
- Einfacher arbeiten mit Kommandofolgen
- Tips für das CLI
- Grundlagen der Amiga-Grafik
- Sounderzeugung mit dem Amiga
- Überblick über Amiga-BASIC

Das »Amiga 500-Buch« stellt eine vollständig aktualisierte und überarbeitete Ausgabe des bereits veröffentlichten und erfolgreichen »Amiga-Handbuches« vom selben Autor dar.

Hardware-Voraussetzung: Commodore Amiga 500

ISB N 3-89090-522-6

Markt&Technik

DM 49,-
sFr 45,10
öS 382,20